UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto

Jamille Silva Madureira

São Cristóvão

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Jamille Silva Madureira

Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Michel dos Santos Soares

Coorientador: Prof. Dr. Rogério Patrício Chagas do Nascimento

São Cristóvão
2017

Jamille Silva Madureira
Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto/ Jamille Silva Madureira. – São Cristóvão, 2017-
87 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Michel dos Santos Soares

Dissertação (Mestrado) – UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, 2017.

1. Avaliação de equipes de desenvolvimento. 2. Métricas de Software. I. Orientador: Prof. Dr. Michel dos Santos Soares. Coorientador: Rogério Patrício Chagas do Nascimento II. Universidade Federal de Sergipe. III. Programa de Pós Graduação em Ciência da Computação. IV. Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto

CDU 02:141:005.7

Jamille Silva Madureira

Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Michel dos Santos Soares
Presidente
Universidade Federal de Sergipe (UFS)

Prof. Dr. Rogério Patrício Chagas do Nascimento
Coorientador
Universidade Federal de Sergipe (UFS)

Prof. Dr. Methanias Colaço Rodrigues Júnior
Membro Interno
Universidade Federal de Sergipe (UFS)

Prof. Dr. Glauco de Figueiredo Carneiro
Membro Externo
Universidade Salvador (UNIFACS)

Avaliação de Equipes de Desenvolvimento de Software por Meio de Métricas Orientadas a Objeto

Este exemplar corresponde à redação da Dissertação de Mestrado, sendo a defesa do mestrando **Jamille Silva Madureira** para ser aprovada pela banca examinadora.

São Cristóvão, 12 de abril de 2017.

Prof. Dr. Michel dos Santos Soares
Orientador

Prof. Dr. Rogério Patrício Chagas do Nascimento
Coorientador

Prof. Dr. Methanias Colaço Rodrigues Júnior
Membro Interno

Prof. Dr. Glauco de Figueiredo Carneiro
Membro Externo

Dedico aos meus queridos avós, Ezequiel e Antônia (in memoriam), os grandes incentivadores dos estudos em toda a família.

Agradecimentos

Ao meu orientador, Michel, sempre dedicado e presente, me direcionando com paciência e carinho em todas as atividades do mestrado. Muito obrigado por tudo!

Ao meu esposo, Ricardo, por ter embarcado comigo nessa aventura ao mudar de estado para que eu pudesse realizar essa conquista.

À minha mãe Diva e meus irmãos Marcelo e Camilla, que sempre foram incentivadores na continuidade dos estudos. Mesmo à distância, o apoio de vocês foi fundamental. Também aos meus sobrinhos queridos, Pedro, Laura e Davi que trouxeram alegria à nossa família. Amo todos vocês.

Aos meus filhos caninos, que sempre souberam como aliviar meu *stress*, e foram tantos nesses dois anos de jornada! Manche, Sol e Lara, mamãe ama vocês!

À minha família, queridos tios, tias, primos e primas, sempre tão apoiadores uns dos outros.

Aos companheiros de mestrado, Anderson, Quelita e Telmo, por compartilhar vitórias e tarefas nesse período tenso de nossas vidas!

Aos professores e funcionários do PROCC pelo convívio e aprendizado. Em especial ao professor Rogério Nascimento pela parceria desenvolvida na coorientação e pelo aprendizado em suas aulas.

Aos colegas do IFS, pelo incentivo e apoio, sempre tão dispostos a me ajudar quando precisei me ausentar.

Aos amigos, incentivadores à distância desse desafio.

A todos vocês, meu muito obrigado!

Resumo

Gerenciar um projeto de software é uma tarefa cada vez mais complexa à medida que as exigências sobre o produto final aumentam. O ambiente competitivo no mercado de software e as necessidades dos clientes exigem que os desenvolvedores de software tenham preocupação cada vez maior na satisfação do usuário como uma medida da qualidade do produto final. Assim, é preciso um gerenciamento eficaz desde a concepção do projeto até a manutenção do software. A equipe de desenvolvimento é um dos recursos mais relevantes para o êxito dos projetos, mas também é onde frequentemente são encontrados os maiores problemas. O uso de métricas é uma forma de ajudar a equipe do projeto a atingir os seus objetivos e metas. O objetivo deste trabalho foi utilizar métricas de software para avaliar equipes e seus membros, analisando o desempenho atual dos desenvolvedores. Para atingir esse objetivo, nesse trabalho foram aplicados como instrumentos de pesquisa a revisão da literatura e o estudo de caso. A revisão da literatura propiciou descobrir valores referência para as métricas aplicadas, com a finalidade de estabelecer critérios que serviram de base para avaliar os softwares envolvidos na pesquisa. Após obter esses conhecimentos, foi aplicado o estudo de caso em duas empresas públicas locais e em exercícios com alunos de graduação de duas universidades. No total, foram avaliados treze softwares, sendo quatro desenvolvidos nas empresas e nove pelos estudantes. Para a aplicação das métricas, foi selecionada uma ferramenta que as coletasse automaticamente e fornecesse o resultado em um formato pronto para ser manipulado. Como resultado, foi constatado que a avaliação da qualidade do software por meio de métricas contribui na gestão de projetos, pois indica suas falhas e onde deve ser melhorado. Também foram descobertas evidências de que o uso de métricas é útil na avaliação dos membros das equipes de desenvolvimento. Por meio do estudo de caso, descobriu-se que a composição da equipe é importante para o sucesso do projeto, pois a formação e experiência dos membros afetam diretamente a qualidade do software. Ao analisar os softwares por meio de métricas, foi observado que o melhor desempenho foi alcançado por desenvolvedores com capacitação em andamento e tempo mais próximo de experiência . Neste sentido, as métricas de software podem contribuir para acompanhar tanto o desenvolvimento do projeto quanto nas decisões que causam mudanças na equipe.

Palavras-chaves: Avaliação, equipes de desenvolvimento, métricas de software orientadas a objeto, Métricas CK, Métricas MOOD, gerenciamento de projetos.

Abstract

Managing a software project is an increasingly complex task as the demands on the final product increase. Competitive environment in software industry and customer needs require software developers to increasingly concern themselves with user satisfaction as a measure of quality of the final product. Therefore, effective management is required from project design up to software maintenance. The development team is one of the most relevant resources for the success of projects, but it is also where the greatest problems are found. Using metrics is a way to help the project team to achieve its goals and objectives. The goal of this work was to use software metrics to evaluate teams and and their members in order to analyze current performance of developers. To reach this objective, literature review and case study were applied as research instruments. The literature review allowed to discover reference values for the applied metrics in order to establish criteria that served as a basis to evaluate the software involved in the research. After obtaining this knowledge, the case study was applied to two local public companies and undergraduate students from two universities. In total, thirteen softwares were evaluated, four of which were developed in companies and nine by students. A tool was selected to collect software metrics automatically and to provide the result in a format ready to be manipulated. As a result, it was verified that the evaluation of software quality by means of metrics contributes to project management, because it indicates its failures and where it should be improved. Evidence has also been discovered that using collected software metrics is useful in evaluating development team members. By means of the case study, it was discovered that team composition is important to the success of the project, and that training and experience of the members directly affect quality of software. When analyzing the software through metrics, it was observed that the best performance was achieved by developers with more advanced training and more near experience time. In this sense, software metrics can contribute to follow both the development of the project and the decisions that cause changes in the team.

Key-words:Evaluation, development teams, object-oriented software metrics, CK metrics, MOOD metrics, project management.

Lista de figuras

Figura 2.1 – Exemplo de Herança 25
Figura 2.2 – Exemplo de Acoplamento 26
Figura 3.1 – Etapas do processo de seleção e análise dos estudos 36
Figura 3.2 – Saída de tela da ferramenta CKJM 40
Figura 3.3 – Saída de tela da ferramenta CCCC 41
Figura 3.4 – Saída de tela da ferramenta JHawk 42
Figura 3.5 – Resultado das métricas CK 42
Figura 3.6 – Resultado das métricas MOOD 43
Figura 3.7 – Resultado das métricas em arquivo *.csv* 43
Figura 5.1 – Métricas vs Qualidade de software 56

Lista de tabelas

Tabela 2.1 – Sumário das métricas CK 27
Tabela 2.2 – Sumário das métricas MOOD 30
Tabela 3.1 – Valores referência para as métricas CK em softwares desenvolvidos na linguagem Java 35
Tabela 3.2 – Trabalhos selecionados 37
Tabela 3.3 – Valores encontrados para Métricas MOOD 38
Tabela 3.4 – Proposta de classificação das métricas 38
Tabela 3.5 – Valores referência para Métricas MOOD 39
Tabela 3.6 – Critérios de escolha das ferramentas 43
Tabela 4.1 – Empresa x Software x Programador 48
Tabela 4.2 – Escolaridade 49
Tabela 4.3 – Experiência em Tecnologia da Informação 49
Tabela 4.4 – Experiência em Programação 49
Tabela 4.5 – Experiência na Linguagem Java 49
Tabela 4.6 – Valores encontrados para as métricas CK nas empresas 50
Tabela 4.7 – Valores encontrados para as métricas MOOD nas empresas 50
Tabela 4.8 – Informações sobre alunos 51
Tabela 4.9 – Valores encontrados para as métricas CK nos softwares desenvolvidos pelos estudantes 52
Tabela 4.10–Valores encontrados para as métricas MOOD nos softwares desenvolvidos pelos estudantes 53
Tabela 5.1 – Uso de Métricas CK em Engenharia de Software 58
Tabela 5.2 – Uso de Métricas MOOD em Engenharia de Software 59
Tabela 5.3 – Valores anômalos encontrados para as métricas CK 61
Tabela 5.4 – Valores anômalos encontrados para as métricas MOOD 63
Tabela 5.5 – Avaliação geral dos softwares 65

Lista de abreviaturas e siglas

AHF	Attribute Hiding Factor
AIF	Attribute Inheritance Factor
CF	Coupling Factor
CK	Chidamber e Kemerer
CBO	Coupling between Object Classes
DIT	Depth of Inheritance Tree
LCOM	Lack of Cohesion in Methods
MHF	Method Hiding Factor
MIF	Method Inheritance Factor
MOOD	Metrics for Object Oriented Design
NOC	Number of Children
PF	Polymorphism Factor
RFC	Response for a Class
WMC	Weighted Methods per Class

Sumário

1	**INTRODUÇÃO**	**14**
1.1	**Contextualização**	**14**
1.2	**Questões de pesquisa**	**15**
1.3	**Objetivos**	**16**
1.4	**Revisão da literatura correlata**	**17**
1.5	**Metodologia**	**19**
1.6	**Estrutura do Trabalho**	**20**
2	**REFERENCIAL TEÓRICO**	**21**
2.1	**Avaliação de desenvolvedores de software**	**21**
2.1.1	**Avaliação individual**	**22**
2.1.2	**Avaliação de equipes**	**23**
2.2	**Métricas de Software**	**23**
2.2.1	**Métricas CK**	**24**
2.2.2	**Métricas MOOD**	**27**
3	**VALORES DE REFERÊNCIA DE MÉTRICAS DE SOFTWARE PARA O PARADIGMA ORIENTADO A OBJETOS**	**31**
3.1	**Valores referência para métricas de software**	**31**
3.2	**Parâmetros para as Métricas CK**	**33**
3.3	**Parâmetros para as Métricas MOOD**	**35**
3.4	**Ferramenta de coleta das métricas**	**39**
4	**ESTUDO DE CASO**	**44**
4.1	**Planejamento**	**44**
4.1.1	**Definição do objetivo**	**44**
4.1.2	**Formulação das suposições**	**45**
4.1.3	**Variável independente**	**45**
4.1.4	**Variáveis dependentes**	**45**
4.1.5	**Variável interveniente**	**46**
4.1.6	**Seleção de participantes**	**46**
4.1.7	**Projeto do experimento**	**47**
4.1.8	**Instrumentação**	**47**
4.2	**Operação do experimento**	**47**
4.2.1	**Preparação**	**47**
4.2.2	**Execução**	**48**

4.3 Resultados 48
4.3.1 Dados coletados nas empresas participantes 48
4.3.2 Dados coletados sobre os alunos de graduação 51
4.4 Ameaças à validade 54

5 AVALIAÇÃO DAS EQUIPES DE DESENVOLVIMENTO 55
5.1 Uso de Métricas orientadas a objetos em atividades de Engenharia de Software 55
5.1.1 Uso das Métricas CK em atividades de Engenharia de Software 56
5.1.2 Uso das Métricas MOOD em atividades de Engenharia de Software . . . 58
5.2 Análise dos resultados 59
5.2.1 Avaliação dos softwares por meio das Métricas CK 59
5.2.2 Avaliação dos softwares por meio das Métricas MOOD 61
5.2.3 Métricas de Software vs Equipe de desenvolvimento 64

6 CONCLUSÃO 67
6.1 Respostas às questões de pesquisa 68
6.2 Contribuições 69
6.3 Trabalhos futuros 70

REFERÊNCIAS 71

APÊNDICES 79

APÊNDICE A – QUESTIONÁRIO APLICADO NA EQUIPE DE DESENVOLVIMENTO 80

APÊNDICE B – GUIA PARA AVALIAÇÃO DE DESENVOLVEDORES 83

1 Introdução

Neste capítulo será mostrada uma visão preliminar do assunto abordado na pesquisa. As seções são apresentadas, nesta ordem: inicialmente, a contextualização na Seção 1.1, em seguida as questões de pesquisa na Seção 1.2, os objetivos do trabalho na Seção 1.3, a revisão da literatura correlata na Seção 1.4, a metodologia aplicada na Seção 1.5 e, por fim, a estrutura do trabalho na Seção 1.6.

1.1 Contextualização

Os computadores estão sendo usados em uma variedade cada vez maior de áreas de aplicação, e seu funcionamento correto é frequentemente crítico para o sucesso do negócio. Desenvolver produtos de software de alta qualidade é de fundamental importância. Um software que atenda às necessidades dos usuários é considerado como um dos principais objetivos da gestão da qualidade (JUNG, 2007), (BEHKAMAL; KAHANI; AKBARI, 2009), (SUKOCO; CUCUS et al., 2012). O gerenciamento eficaz é um dos fatores cruciais que possibilitam o desenvolvimento bem-sucedido de produtos de alta qualidade (NAIR; SUMA; TIWARI, 2012). De acordo com a norma ISO 25000, um software é de qualidade quando é utilizado sob condições específicas e apresenta a capacidade de satisfazer tanto às necessidades declaradas, como também as não declaradas (ISO, 2005).

A equipe de desenvolvimento de software tem sido reconhecida como um dos recursos mais decisivos para o sucesso dos projetos, mas também a principal fonte de deficiências. O gerente de projetos precisa mostrar preocupação pelas pessoas, construir confiança, mostrar simpatia e envolver as emoções das pessoas, por exemplo, na resolução de problemas (FISHER, 2011), (COLOMO-PALACIOS et al., 2012). O desempenho da equipe em projetos de desenvolvimento de software é dependente da interação de fatores diferentes, tais como: planos eficazes, boa comunicação, metas claras, dentre outros. A avaliação do esforço individual e da equipe pelo gerente tem influência no nível de qualidade do projeto (RYAN; O'CONNOR, 2009), (NAIR; SELVARANI, 2012).

A consciência do esforço exigido por uma equipe de desenvolvimento é um dos fatores que influenciam para alcançar a qualidade do processo. O desempenho do indivíduo tende a crescer com o aumento do conhecimento do domínio da aplicação e de seus conjuntos de habilidades (NAIR; SELVARANI, 2012). Os métodos aplicados na análise do desempenho da equipe devem ser projetados para avaliar tanto o processo de desenvolvimento, quanto os resultados. Para analisar os resultados, os critérios utilizados são a aderência ao cronograma e ao orçamento estimado, a produtividade e a qualidade do software produzido (ROSEN et al., 2010), (LALSING; KISHNAH; PUDARUTH, 2012).

Uma das maneiras de avaliar a qualidade do software é por meio de métricas (WALWORTH; YEARWORTH; SHRIEVES, 2014). Quando aplicadas corretamente, as métricas proporcionam diversos benefícios dentre os quais a avaliação eficaz da otimização de processos de desenvolvimento de software e o apoio ao gerente de projeto na tomada de decisões estratégicas (MATUSSE; HUZITA; TAIT, 2012).

A norma IEEE 1061 estabelece uma metodologia para o desenvolvimento de métricas para atributos de qualidade de software. A norma define um atributo como uma característica física ou abstrata de uma entidade que pode ser medida. Uma métrica de qualidade de software é uma função cujas entradas são os dados de software e a saída é um único valor numérico que pode ser interpretado como o grau em que o software possui um determinado atributo que afeta a sua qualidade. Métricas de projeto de software visam medir e avaliar as características de desempenho de um software e podem ser utilizadas para identificar os defeitos do produto e avaliar a qualidade do software (IQBAL; NAEEM; KHAN, 2012).

As métricas de software permitem a medição, avaliação, controle e melhoria de produtos e processos de software. Seu uso contribui para que um gerente possa avaliar se o projeto está progredindo de acordo com o que foi planejado. Métricas também são úteis para determinar o *status* atual de um projeto e avaliar o seu desempenho. A identificação precoce dos riscos associados às diferentes atividades do projeto oferece uma oportunidade para concentrar esforços nas tarefas mais críticas (FERREIRA et al., 2012), (IQBAL; NAEEM; KHAN, 2012).

Neste trabalho, as métricas de software foram usadas para avaliar equipes e indivíduos, analisando o desempenho atual e perspectivas de desempenho futuro do ponto de vista gerencial em um projeto de desenvolvimento de software.

1.2 Questões de pesquisa

As questões de pesquisa estabelecem o que é necessário saber para cumprir o objetivo do estudo (RUNESON; HÖST, 2009).

De acordo com a pesquisa descrita em (PERRY; SIM; EASTERBROOK, 2004), espera-se que um estudo de caso tenha questões de pesquisa estabelecidas desde o início, os dados sejam coletados de forma planejada e consistente, inferências sejam feitas a partir dos dados para responder à questão de pesquisa e ameaças à validade sejam abordadas.

A fim de cumprir esses propósitos, o estudo de caso realizado neste trabalho foi planejado de acordo com as diretrizes apresentadas em (WOHLIN et al., 2012). A execução do estudo de caso está descrita no Capítulo 4 deste documento, desde o planejamento até os resultados obtidos.

Esta pesquisa está embasada na premissa de que **"Métricas de software do paradigma orientado a objetos podem ser usadas na avaliação técnica da equipe de desenvolvimento de software".**

Para avaliar esta proposição, as seguintes questões de pesquisa foram elaboradas:

Q1: Métricas de software orientadas a objetos podem ser usadas para avaliar a qualidade do produto final, do ponto de vista do gerenciamento de projetos?

Q2: É possível avaliar tecnicamente equipes de desenvolvimento de software e seus membros por meio de métricas de produto de software?

As questões de pesquisa são respondidas no Capítulo 5, onde é feita a análise dos resultados obtidos na realização do estudo de caso.

1.3 Objetivos

Gerenciar um projeto de software é uma tarefa que se torna cada vez mais difícil à medida que a complexidade do software aumenta (NIENABER; BARNARD, 2007). Os gerentes são obrigados a administrar cada passo do processo de execução do projeto para que as tarefas possam ser realizadas dentro dos parâmetros acordados. O uso de métricas é uma forma de ajudar a equipe do projeto a atingir os seus objetivos e metas (IQBAL; NAEEM; KHAN, 2012).

Este trabalho teve como objetivo geral utilizar métricas para avaliar produtos de software desenvolvidos de acordo com o paradigma orientado a objeto. Em seguida, os resultados foram aplicados para avaliar o desempenho da equipe de desenvolvimento.

Para que o objetivo geral fosse alcançado, os seguintes objetivos específicos foram realizados:

1. Identificar padrões de comportamento inadequados de desenvolvimento de software orientado a objetos usando dados obtidos na literatura;
2. Avaliar de que forma as métricas de software são usadas para identificar padrões inadequados na equipe de desenvolvimento;
3. Analisar a aplicabilidade de métricas para avaliar a qualidade do produto final do ponto de vista do gerenciamento de projetos;

4. Identificar métricas que podem ser usadas na determinação de capacidade individual e da equipe de desenvolvimento de software;

5. Avaliar a conformidade de indivíduos e equipes de desenvolvimento de software em relação à adequação ao desenvolvimento de software orientado a objetos usando métricas específicas.

1.4 Revisão da literatura correlata

No trabalho descrito em (SURESH; PATI; RATH, 2012) é mostrada a avaliação de um software utilizando métricas específicas para o paradigma da orientação a objetos. Os conjuntos de métricas analisados foram McCabe (MCCABE, 1976), Métricas de Martin (MARTIN, 1994) e CK (CHIDAMBER; KEMERER, 1994). Os autores concluíram que o conjunto CK foi considerado o melhor indicador de propensão a falhas. Foi observado que, a partir de inferência de métricas CK, é possível prever a confiabilidade do sistema com mais precisão. Estas métricas ajudam os programadores e testadores a tomar melhor decisão de projeto e também para estimar o esforço de teste. Com exceção da métrica LCOM, todas as métricas do conjunto CK foram eficazes em prever propensão a falha de um sistema.

Na pesquisa apresentada em (IQBAL; NAEEM; KHAN, 2012), os autores avaliaram como diferentes métricas relacionadas às diferentes áreas do projeto de software, especialmente em engenharia de requisitos, podem ser úteis para gerenciar os projetos de software. Os autores constataram que essas métricas são muito eficazes em informar sobre o estado atual dos diferentes processos de software. Os pesquisadores concluíram que, ao adotar métricas de requisitos, as empresas de desenvolvimento de software podem se beneficiar de uma visão mais profunda sobre seus projetos, que não só irá ajudar a melhor administrar o projeto de software, mas também contribuir para o alcance dos objetivos de negócio de forma eficaz.

No trabalho relatado em (ROBLES et al., 2012), métricas foram propostas para avaliar a evolução das funções com base no número de vezes que elas foram modificadas. As métricas foram aplicadas em dois projetos de software livre, um na comunidade Apache e o outro na empresa Novell Evolution. Os autores concluíram que a análise da evolução do software feita sobre sua semântica (funções) em vez de no nível físico (arquivo, linha) permite ganhar uma compreensão maior sobre o projeto de software em estudo. Por meio desse estudo, os autores verificaram que a maioria das funções raramente muda, e quando há modificações, isto geralmente acontece na fase de concepção. Outra observação realizada foi que o padrão de mudança de funções depende muito da participação contínua de seu autor no projeto de software, o que retrata a importância do fator humano no campo da evolução e manutenção de software.

No estudo apresentado em (SHARMA; KALIA; SINGH, 2012), é feito um levantamento das métricas Lorenz e Kidd (LORENZ; KIDD, 1994), CK (CHIDAMBER; KEMERER, 1994) e MOOD (ABREU; CARAPUÇA, 1994). Os autores concluíram que a suíte CK foi influente na

definição de métricas e validação de qualidade. Da mesma forma, a suíte MOOD é bem definida, o nível de projeto é matematicamente calculável e fornece limiares que podem ser usados para julgar as métricas coletadas a partir de um determinado projeto. As métricas de Lorenz e Kidd são medidas estatísticas de software em termos de contagem, mas são ineficazes para medir a qualidade do software. Os autores recomendam métricas que são úteis na avaliação da qualidade de software. Do conjunto CK, as métricas WMC, RFC, DIT, NOC e CBO são adequados para avaliação da qualidade de software, e a partir da suíte MOOD as métricas adequadas são MHF, AHF, MIF, AIF e PF. Estas dez métricas são úteis em calcular encapsulamento, herança e polimorfismo.

Na pesquisa apresentada em (GUPTA; BATRA et al., 2012), os autores explicam que as métricas ajudam na identificação de adversidades oriundas do processo de desenvolvimento de software. Eles afirmam que encontrar esses problemas na fase em que são desenvolvidos diminui o custo e evita efeitos de ondulação importantes em fases posteriores de desenvolvimento. Os pesquisadores reconhecem que, atualmente, o paradigma orientado a objetos é uma importante área de pesquisa em Engenharia de Software. Os autores concluem que nenhum conjunto de métricas é completo e que não existe uma única métrica que possa medir todos os aspectos de um sistema orientado a objeto.

No estudo descrito em (JASSIM; ALTAANI, 2013), os pesquisadores afirmam que o paradigma orientado a objetos está se tornando mais popular no desenvolvimento de software e que as métricas deste paradigma são uma parte essencial do ambiente de desenvolvimento de software. Um modelo de regressão linear é usado para encontrar a relação entre os fatores do método MOOD e suas influências nas medições de software orientado a objeto. Os autores concluíram que tais medições permitem aos projetistas avaliar o software desde o início do processo, fazendo alterações que reduzam a complexidade e melhoram a capacidade contínua do projeto.

O trabalho apresentado em (WALWORTH; YEARWORTH; SHRIEVES, 2014) explica que a necessidade de medição para permitir um controle eficaz em projetos de software é bem reconhecida. É preciso compreender a progressão técnica de projetos, especialmente onde os projetos têm altos níveis de complexidade. As investigações evidenciam a dificuldade tanto na implantação e utilização de um regime de medição, quanto na divulgação dos resultados para a organização. A administração das medições seria útil para as organizações, pois ajudaria a controlar o gerenciamento de funções. Na pesquisa é apresentada uma proposta para a criação de uma ferramenta KM (*Knowledge Management*) que permitirá o controle dos projetos, coletando, armazenando e exibindo as informações para o público desejado.

Como pode ser observado, a aplicação de métricas é útil para monitorar o processo de desenvolvimento de software. As métricas ajudam a identificar problemas e a partir disso tomar decisões que possam colaborar com o progresso do projeto. Neste trabalho, as métricas serão utilizadas para avaliar a equipe de desenvolvimento por meio do software produzido.

1.5 Metodologia

No intuito de alcançar os objetivos propostos, nesse trabalho foram aplicados a revisão da literatura e o estudo de caso como instrumentos de pesquisa.

O objetivo de realizar a revisão da literatura foi buscar trabalhos que ajudassem a adquirir conhecimentos sobre métricas específicas de projetos orientados a objeto e como essas métricas podem ser usadas em seu gerenciamento. As buscas foram restritas aos últimos vinte anos, consultadas nas bases digitais *IEEExplore, ACM, Scopus e ScienceDirect*.

Por meio dessa revisão, descobriu-se que métricas de software são aplicadas para acompanhar o processo de desenvolvimento de software, porém o foco da supervisão é na qualidade do produto e no cumprimento de prazos e custos. Neste trabalho, a aplicação de métricas de software irá ajudar na tomada de decisão no gerenciamento de projetos ao considerar a qualificação técnica dos desenvolvedores.

A equipe de desenvolvimento foi avaliada de acordo com a qualidade do software produzido, considerando características apresentadas em (BENESTAD; ANDA; ARISHOLM, 2006), (ZHOU; LEUNG, 2006), (OLBRICH et al., 2009), (GANDHI; BHATIA, 2010), (JOHARI; KAUR, 2012) e (SRIVASTAVA; KUMAR, 2013). Essas características incluem: manutenibilidade, testabilidade, compreensibilidade, reutilização, eficiência e propensão a erros.

Foram buscados na literatura valores referência para as métricas aplicadas, com a finalidade de obter critérios que sirvam de base para fazer a comparação em relação a outros softwares desenvolvidos. Para que as métricas fossem utilizadas de maneira eficaz, foi buscada uma ferramenta de software que aplicasse as métricas selecionadas e gerassem o resultado automaticamente.

Também foi utilizado como instrumento de pesquisa a execução de um estudo de caso. Pesquisa sobre engenharia de software tem como objetivo investigar como o desenvolvimento, operação e manutenção de software são realizados por engenheiros de software e outras partes interessadas. O desenvolvimento de software é realizado por indivíduos, grupos e organizações, e questões sociais e políticas são de importância para este desenvolvimento. Assim, a engenharia de software é uma área multidisciplinar que envolve áreas onde estudos de casos normalmente são realizados (RUNESON; HÖST, 2009).

Duas empresas públicas locais propiciaram que o estudo de caso fosse aplicado em sua equipe de desenvolvimento de software, concedendo seis desenvolvedores e quatro softwares produzidos pelos mesmos. O estudo também foi aplicado para execução por alunos de graduação de uma universidade pública e outra particular. As métricas descritas na Seção 2.2 do Capítulo 2 foram aplicadas em softwares desenvolvidos nos locais do estudo.

Os resultados foram analisados se estão dentro dos parâmetros já relatados na literatura, conforme apresentado no Capítulo 3. Neste sentido, a equipe foi avaliada tecnicamente por meio do software desenvolvido, permitindo que os gerentes de projeto tomassem decisões futuras sobre a equipe (por exemplo: necessidade de treinamentos específicos, aumento de salários, mudanças de cargo) com base no desempenho técnico dos seus membros.

1.6 Estrutura do Trabalho

Esta dissertação está organizada em 6 capítulos. Uma breve descrição do conteúdo de cada capítulo é apresentada a seguir.

No Capítulo 2 são apresentados conteúdos relevantes para a realização deste trabalho. Inicialmente, os conceitos sobre avaliação de desenvolvedores de software são explanados. Em seguida, as métricas de software utilizadas nesta pesquisa são explicadas.

No Capítulo 3 são apresentados como foram buscados na literatura valores referência para as métricas aplicadas. Esses valores tem a finalidade de obter critérios que sirvam de base para fazer a avaliação dos softwares utilizados nesta pesquisa. Também é abordado como estes valores podem contribuir na gestão de projetos. Por fim, a seleção da ferramenta utilizada para a coleta de métricas é mostrada.

No Capítulo 4 são apresentados o planejamento, a execução e os resultados do estudo de caso realizado nesta pesquisa.

No Capítulo 5 é apresentada a avaliação de equipes de desenvolvimento por meio de métricas de software. Inicialmente, será explanado como as métricas do paradigma orientado a objetos podem contribuir em atividades da Engenharia de Software. Em seguida, será abordado como as métricas CK e MOOD podem colaborar nessas atividades. Por último, são discutidos os resultados do estudo de caso.

Por fim, no Capítulo 6 são apresentadas as respostas às questões da pesquisa, as contribuições do trabalho e as sugestões de trabalhos futuros.

2 Referencial teórico

Neste capítulo, são apresentados conteúdos relevantes para a realização deste trabalho. A Seção 2.1 refere-se aos conceitos sobre avaliação de desenvolvedores de software. Na Subseção 2.1.1, é explanado sobre a avaliação individual, na Subseção 2.1.2, é abordada a avaliação de equipes de desenvolvimento de software. Na Seção 2.2, as métricas de software utilizadas nesta pesquisa são explicadas. As Subseções 2.2.1 e 2.2.2 referem-se ao conjunto de métricas CK e MOOD, respectivamente.

2.1 Avaliação de desenvolvedores de software

A gestão de pessoas desempenha um papel importante na gestão de projetos (FISHER, 2011). Pesquisas recentes afirmam que os recursos humanos são a chave para o desenvolvimento de software, sugerindo que o potencial humano é o recurso mais importante (GALINEC, 2010), (COLOMO-PALACIOS et al., 2012), (YILMAZ et al., 2017). O desenvolvimento de software depende significativamente do desempenho da equipe, assim como qualquer processo que envolve a interação humana (MOE; DINGSØYR; DYBÅ, 2010).

Os membros das equipes de desenvolvimento de software são caracterizados como indivíduos com habilidade especializada, capazes de aplicar essas habilidades para identificar e resolver problemas. O conhecimento, especialmente o tácito, é mantido individualmente e é o meio de produção no desenvolvimento de software. O processo de desenvolvimento envolve a coordenação tácita da experiência desses membros da equipe (FARAJ; SPROULL, 2000), (RYAN; O'CONNOR, 2009) .

Estudos sobre grupos e equipes indicam um conjunto de variáveis que estão associadas com a eficácia da equipe. Estas variáveis incluem características de membros da equipe (personalidade, habilidade, experiência) e elementos do contexto organizacional que suportam a eficácia da equipe, tais como treinamento, composição da equipe, diversidade de membros e comunicação (HYATT; RUDDY, 1997), (JORDAN; FEILD; ARMENAKIS, 2002), (CARPENTER, 2002), (LEPINE, 2003).

O comportamento dos membros da equipe também foi estudado. Por exemplo, a eficácia da equipe pode ser reduzida pelo comportamento social, que é uma tendência a exercer menos esforço quando se trabalha em grupo do que quando se trabalha individualmente (LOUGHRY; OHLAND; MOORE, 2007).

Uma parte importante da gestão de recursos humanos é conhecer efetivamente os membros da equipe e saber atribuir às pessoas seus respectivos papéis nos projetos. Este processo é crucial para a geração de equipes produtivas e ajuda a desenvolver competência sistemática a

longo prazo (ACUÑA; JURISTO; MORENO, 2006).

Portanto, ao avaliar a equipe de desenvolvimento, torna-se possível saber os pontos positivos e onde há necessidade de melhorias. Este conhecimento é fundamental para que as empresas possam gerenciar seus projetos de maneira eficaz.

2.1.1 Avaliação individual

A produção de software envolve o elemento humano, exigindo atividades como resolução de problemas, pensamento analítico e comunicação. Além disso, o desenvolvimento de software compreende diferentes atividades, como análise de sistemas, projeto, codificação e testes. Para executar essas atividades, é necessário que os membros da equipe possuam não somente diversas habilidades, mas também diferentes tipos de personalidade (AHMED; CAPRETZ; CAMPBELL, 2012). Ao avaliar características individuais, pesquisadores propuseram grupos de comportamentos e atributos importantes para os membros da equipe.

Na pesquisa descrita em (STEVENS; CAMPION, 1999), os autores sugerem que existem catorze requisitos, denominados de KSA's (*The Knowledge, Skill and Ability*), para que o indivíduo consiga trabalhar em equipe. Estes KSA's são divididos em cinco grupos: (1) resolução de conflitos, (2) solução de problemas de maneira colaborativa, (3) comunicação, (4) definição de objetivos e gestão do desempenho, (5) planejamento e coordenação de tarefas. Cada grupo contém as habilidades específicas para que o indivíduo consiga atender essas atividades. A pesquisa inclui um teste para avaliar esses requisitos.

Outra abordagem de avaliação é associar as tarefas desenvolvidas com a personalidade dos membros da equipe. Como exemplo, pode-se citar a pesquisa relatada em (GÓMEZ; ACUÑA, 2007), onde os autores analisaram as relações entre a personalidade, características das tarefas e a qualidade do produto. Foi observado que traços como sociabilidade, comunicabilidade, afabilidade e abertura são propícios para o desenvolvimento de software com alta qualidade, bem como para a satisfação dos membros da equipe. Os autores também constataram que a extroversão deve ser considerada como um preditor válido de qualidade de software quando são adotadas as metodologias ágeis no processo de desenvolvimento, pois a alta interação entre os membros da equipe é essencial para este método de desenvolvimento.

Outro exemplo dessa abordagem de avaliação pode ser observado no trabalho apresentado em (WIESCHE; KRCMAR, 2014). Os autores concluíram que há características que podem ajudar a definir a melhor personalidade para as tarefas propostas a um desenvolvedor. Eles descobriram que as tarefas que exigem desenvolvimento individual e criatividade combinam com pessoas de personalidade introvertida, enquanto as tarefas que exigem colaboração, como programação em pares, combinam com pessoas de personalidade extrovertida.

2.1.2 Avaliação de equipes

Diversas pesquisas (LEPINE et al., 2008), (HÜLSHEGER; ANDERSON; SALGADO, 2009), (RYAN; O'CONNOR, 2009), (KOSTOPOULOS; BOZIONELOS, 2011), (DINGSØYR; DYBÅ, 2012), (HODA; NOBLE; MARSHALL, 2013) indicam a importância do trabalho em equipe para o sucesso de projetos. O desempenho real da equipe é complexo e depende não somente da competência dos seus membros na gestão e execução de suas atividades, mas também no contexto organizacional fornecido pela administração (MOE; DINGSØYR; DYBÅ, 2010).

Uma maneira de avaliar equipes foi descrita em (GUZZO; DICKSON, 1996). Os autores sugerem que três fatores influenciam na eficácia de uma equipe: resultados produzidos em grupo (quantidade ou qualidade, velocidade, satisfação do cliente, dentre outros), os efeitos que um grupo causa para seus membros e a capacidade de aprimoramento de uma equipe para o futuro.

Nesse sentido, o desempenho da equipe pode ser definido como a medida em que uma equipe é capaz de cumprir a qualidade estabelecida e os objetivos de custo e tempo. O desempenho depende da interação de fatores, tais como planos eficazes, boa comunicação e objetivos claros. A percepção do sucesso do projeto depende, em parte, da perspectiva do avaliador. Assim, é importante incluir múltiplos aspectos (empresa, cliente, equipe) ao avaliar o desempenho da equipe (HOEGL; GEMUENDEN, 2001), (RYAN; O'CONNOR, 2009).

Capturar completamente o conjunto de processos que compreendem o desempenho da equipe envolve múltiplas abordagens para a medição. Os métodos e métricas utilizados para a avaliação do desempenho da equipe devem ser projetados para capturar tanto o processo da equipe quanto os resultados (ROSEN et al., 2010).

Consequentemente, para avaliar a equipe de software são considerados tanto os resultados psicossociais quanto os resultados das tarefas. Os resultados psicossociais da equipe referem-se à avaliação do grau de sentimentos positivos associados às interações, conhecimento e habilidades adquiridas, satisfação na participação no processo de implementação do projeto. Os resultados de tarefas de projeto de software normalmente se referem à aderência ao cronograma, ao orçamento estimado e à produtividade em termos de instruções de código-fonte e, por fim, à qualidade do software produzido (ANDRES, 2002), (LALSING; KISHNAH; PUDARUTH, 2012).

Neste trabalho, as equipes serão avaliadas por meio da qualidade do software produzido. Para isso, serão usadas métricas de software orientadas a objetos, explanadas na Seção 2.2.

2.2 Métricas de Software

Medição de software é o processo de representação em números de entidades de software, como processos, produtos e recursos (RUNESON; HÖST, 2009). Dentre aquelas que avaliam o produto, pode-se citar métricas para o modelo de requisitos, para o código fonte e também para o modelo do projeto. Neste trabalho, o foco são as métricas para projeto orientado a objetos.

Métricas orientadas a objetos tornaram-se importantes devido à crescente popularidade deste paradigma como a base para a maioria dos sistemas de software (EL-LATEEF; YOUSEF; ISMAIL, 2008). Em uma pesquisa realizada em 2013 (RADJENOVIĆ et al., 2013), foi apresentado que métricas orientadas a objetos (49%) foram utilizadas quase duas vezes mais frequentemente em comparação com as métricas tradicionais de código fonte (27%) ou métricas de processo (24%). Assim, neste trabalho foram consideradas apenas métricas orientadas a objeto.

Para avaliar softwares desenvolvidos no paradigma orientado a objeto, um conjunto de características deve ser observado, tais como: tamanho, complexidade, acoplamento, totalidade, coesão, similaridade e volatilidade (WHITMIRE, 1997).

Neste trabalho, os conjuntos de métricas CK (Chidamber and Kemerer) e MOOD (*Metrics for Object Oriented Design*) foram aplicados para avaliar a qualidade dos softwares produzidos pelas equipes.

2.2.1 Métricas CK

As métricas CK (CHIDAMBER; KEMERER, 1994) foram definidas por Chidamber e Kemerer em 1994. O objetivo das métricas é medir a complexidade do projeto desenvolvido no paradigma orientado a objeto em relação ao seu impacto sobre atributos de qualidade, como usabilidade, facilidade de manutenção, funcionalidade e confiabilidade. Seis métricas foram definidas e são brevemente explicadas a seguir: *Weighted Methods Per Class*, *Depth of Inheritance Tree*, *Number of Children*,*Coupling between Object Classes*, *Response For a Class* e *Lack of Cohesion in Methods* (DUBEY; RANA, 2010).

1. WMC (*Weighted Methods Per Class*):

A métrica WMC avalia a complexidade de uma classe, desconsiderando os métodos herdados. É calculada realizando a soma das complexidades de todos os métodos definidos em uma classe (DUBEY; RANA, 2010)(ELISH; AL-YAFEI; AL-MULHEM, 2011).

Pode ser definida como:

$$WMC = \sum_{i=1}^{n} Ci \tag{1}$$

onde n é o número de métodos definidos na classe e Ci a complexidade do método i (GANDHI; BHATIA, 2010).

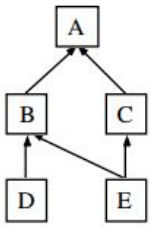
Complexidade não é definida pelos autores, com o objetivo de permitir a aplicação mais geral desta métrica. É importante ressaltar que qualquer métrica de complexidade utilizada deverá ter as propriedades de uma escala de intervalo para permitir o somatório (CHIDAMBER; KEMERER, 1994). Como exemplos de medidas de complexidade, pode-se citar o número

ciclomático de McCabe (MCCABE, 1976), o esforço de programação de Halstead (HALSTEAD, 1977) e as medidas de Ponto de Função (ALBRECHT; GAFFNEY, 1983).

2. DIT (*Depth of Inheritance Tree*):

A métrica DIT é definida como o caminho máximo de herança da classe para a classe raiz. Nos casos que envolvem a herança múltipla, DIT será o comprimento máximo a partir do nó de raiz da árvore (CHIDAMBER; KEMERER, 1994),(KAUR et al., 2010).

Figura 2.1 – Exemplo de Herança



Fonte: (SHELDON; JERATH; CHUNG, 2002)

Considerando a árvore de herança da Figura 2.1, pode-se afirmar que DIT(A) = 0, pois A é a classe raíz. DIT (B) = DIT (C) = 1, visto que o comprimento destas classes para a classe A é apenas um. Por fim, DIT (D) = DIT (E) = 2, uma vez que o comprimento destas classes para a classe A é dois (SHELDON; JERATH; CHUNG, 2002).

3. NOC (*Number of Children*):

A métrica NOC determina o número de subclasses diretas de uma determinada classe, avaliando a amplitude da sua hierarquia. Quanto maior o valor do NOC, menor o número de falhas. Essa métrica avalia principalmente a eficiência, reutilização e capacidade de teste (GANDHI; BHATIA, 2010).

Ainda considerando a árvore de herança representada na Figura 2.1, pode-se afirmar que NOC(A) = NOC (B) = 2 porque o número de filhos imediatos é igual a dois. NOC (C) = 1, visto que o número de filhos imediatos é igual a um. Finalmente NOC(D) = NOC (E) = 0, pois essas classes não possuem classes filhas (SHELDON; JERATH; CHUNG, 2002).

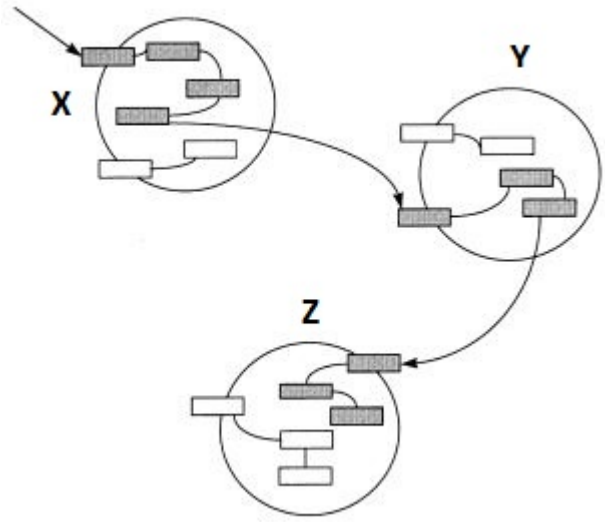
4. CBO (*Coupling between Object Classes*):

A métrica CBO é definida como o número de acoplamentos entre uma determinada classe com todas as outras classes. Duas classes estão acopladas quando os métodos declarados em uma classe usa métodos ou instancia variáveis definidas por outras classes (ELISH; AL-YAFEI; AL-MULHEM, 2011), (GANDHI; BHATIA, 2010).

No nível mais básico, diferentes classes podem ser acopladas uma à outra por ligações diretas através de ponteiros embutidos. Em seguida, dentro de uma classe, podem ser acopladas várias funções entre si. Como ilustrado na Figura 2.2, onde a classe X está diretamente acoplada a

Y e Y também está diretamente acoplada a Z portanto X é indiretamente acoplada a Z (WILKIE; KITCHENHAM, 2000).

Figura 2.2 – Exemplo de Acoplamento



Fonte: Adaptado de (WILKIE; KITCHENHAM, 2000)

5. RFC (*Response For a Class*):

A métrica RFC é definida como um conjunto de métodos que podem ser executados em resposta a uma mensagem recebida por um objeto dessa classe. A métrica inclui todos os métodos acessíveis dentro da hierarquia da classe. É feita uma análise da combinação da complexidade de uma classe por meio do número de métodos e a quantidade de comunicação com outras classes (GANDHI; BHATIA, 2010), (KAUR et al., 2010).

A responsabilidade de uma classe pode ser expressa por:

$$RS = \{M\} \bigcup all\ i\ \{R_i\} \tag{2}$$

onde M é o conjunto de todos os métodos da classe e R_i é o conjunto de métodos chamado pelo método i (KAUR et al., 2010).

6. LCOM (*Lack of Cohesion in Methods*):

A métrica LCOM indica a falta de coesão de métodos. Considerando a classe C1 com n métodos M1, M2,..., Mn. Seja I_j o conjunto de variáveis instanciadas pelo método M_i. Existem n conjuntos I1,I2... In. Seja

$$P = \{(Ii, Ij) \mid Ii \cap Ij \neq \emptyset\}. \tag{3}$$

Se todos os conjuntos I1,I2... In são vazios, então P é vazio (CHIDAMBER; KEMERER, 1994) (KAUR et al., 2010) .

A métrica LCOM pode ser definida como:

$$LCOM = \mid P \mid - \mid Q \mid, se \mid P \mid > \mid Q \mid$$
$$LCOM = 0, para \mid P \mid \leq \mid Q \mid \quad (4)$$

Na Tabela 2.1 é apresentado um sumário das métricas CK, indicando o seu escopo e quais aspectos são avaliados por meio delas.

Tabela 2.1 – Sumário das métricas CK

Métrica	**Avaliação**	
	Característica	**Escopo**
DIT	Herança	Classe
NOC	Herança	Classe
CBO	Acoplamento	Classe
RFC	Acoplamento	Classe
LCOM	Coesão	Classe
WMC	Complexidade	Classe

2.2.2 Métricas MOOD

O conjunto de métricas MOOD (ABREU; CARAPUÇA, 1994) foi proposto por Abreu e Carapuça em 1994 para medir o uso de métodos de projetos desenvolvidos no paradigma orientado a objeto. Seis métricas foram definidas e são brevemente introduzidas a seguir: *Method Inheritance Factor*, *Attribute Inheritance Factor*, *Method Hiding Factor*, *Attribute Hiding Factor*, *Polymorphism Factor* e *Coupling Factor* (DUBEY; RANA, 2010).

1. MIF (*Method Inheritance Factor*):

A métrica MIF é definida como um quociente entre a soma dos métodos herdados em todas as classes do sistema e o número total de métodos disponíveis (definidos localmente e herdados) para todas as classes (GENERO; PIATTINI; CALERO, 2005):

$$MIF = \frac{\sum_{i=1}^{TC} Mi(Ci)}{\sum_{i=1}^{TC} Ma(Ci)} \quad (5)$$

onde a somatória inicia com *i* igual a 1 e finaliza até atingir o número total de classes **TC** e:

$$Ma(Ci) = Md(Ci) + Mi(Ci) \quad (6)$$

Ma(Ci) = número de métodos que podem ser invocados em associação com Ci;

Md(Ci) = número de métodos declarados na classe Ci;

Mi(Ci) = número de métodos herdados (e não cancelados) em Ci.

2. AIF (*Attribute Inheritance Factor*):

A métrica AIF é definida como um quociente entre a soma dos atributos herdados em todas as classes do sistema e o número total de atributos disponíveis (definidos localmente somados aos herdados) para todas as classes (GENERO; PIATTINI; CALERO, 2005):

$$AIF = \frac{\sum_{i=1}^{TC} Ai(Ci)}{\sum_{i=1}^{TC} Aa(Ci)} \tag{7}$$

onde a somatória inicia com *i* igual a 1 e finaliza até atingir o número total de classes **TC** e:

$$Aa(Ci) = Ad(Ci) + Ai(Ci) \tag{8}$$

Aa(Ci) = número de atributos disponíveis na classe Ci;

Ad(Ci) = número de atributos definidos na classe Ci;

Ai(Ci) = número de atributos herdados na classe Ci.

3. MHF (*Method Hiding Factor*):

A métrica MHF é calculada pela razão entre a quantidade de métodos ocultos em todas as classes do sistema e a quantidade total de métodos definidos em todas as classes do sistema (DUBEY; SHARMA et al., 2012).

MHF é definida por (ABREU; CARAPUÇA, 1994):

$$MHF = \frac{\sum_{i=1}^{TC} Mh(Ci)}{\sum_{i=1}^{TC} Md(Ci)} \tag{9}$$

onde a somatória inicia com *i* igual a 1 e finaliza até atingir o número total de classes **TC** e:

$$Md(Ci) = Mv(Ci) + Mh(Ci) \tag{10}$$

Md(Ci) = número de métodos definidos na classe Ci;

Mv(Ci) = número de métodos vísiveis na classe Ci;

Mh(Ci) = número de métodos ocultos na classe Ci.

4. AHF (*Attribute Hiding Factor*):

A métrica AHF é calculada pela divisão entre a quantidade de atributos ocultos em todas as classes do sistema e a quantidade total de atributos definidos em todas as classes do sistema (KRISHNAIAH; PRASAD, 2012).

AHF é definida por (ABREU; CARAPUÇA, 1994):

$$MHF = \frac{\sum_{i=1}^{TC} Ah(Ci)}{\sum_{i=1}^{TC} Ad(Ci)} \tag{11}$$

onde a somatória inicia com *i* igual a 1 e finaliza até atingir o número total de classes **TC** e:

$$Ad(Ci) = Av(Ci) + Ah(Ci) \tag{12}$$

Ad(Ci) = número de atributos definidos na classe Ci;

Av(Ci) = número de atributos vísiveis na classe Ci;

Ah(Ci) = número de atributos ocultos na classe Ci.

5. PF (*Polymorphism Factor*):

A métrica PF é definida como o quociente entre o real número de possíveis diferentes situações polimórficas e o número máximo de possíveis situações polimórficas distintas por classe (DUBEY; SHARMA et al., 2012).

É calculada por:

$$PF = \frac{\sum_{i=1}^{TC} Mo(Ci)}{\sum_{i=1}^{TC} [Mn(Ci\ x\ DC(Ci)]} \tag{13}$$

onde a somatória inicia com *i* igual a 1 e finaliza até atingir o número total de classes **TC** e:

Mo(Ci) = métodos sobrescritos na classe Ci,

Mn(Ci) = novos métodos na classe Ci,

DC(Ci) = números de descendentes da classe Ci (GENERO; PIATTINI; CALERO, 2005).

6. CF (*Coupling Factor*):

A métrica CF é definida como a razão entre o número máximo possível de acoplamentos no sistema para o número real de acoplamentos não atribuídos à herança (DUBEY; SHARMA et al., 2012), (KRISHNAIAH; PRASAD, 2012).

É calculada por:

$$CF = \sum i \sum j \text{ is client} \frac{(C_i, C_j)}{T_C^2 - T_c} \tag{14}$$

onde a somatória de $i = 1$ a Tc e $j = 1$ a Tc. A função *is client* = 1, se, e somente se, existir uma relação entre a classe cliente Cc e a classe servidora Cs, e Cc seja diferente de Cs, caso contrário, será zero (ABREU; CARAPUÇA, 1994).

Na Tabela 2.2 é apresentado um sumário das métricas CK, indicando o seu escopo e quais aspectos são avaliados por meio delas.

Tabela 2.2 – Sumário das métricas MOOD

Métrica	**Avaliação**	
	Característica	**Escopo**
MIF	Herança	Sistema
AIF	Herança	Sistema
MHF	Encapsulamento	Sistema
AHF	Encapsulamento	Sistema
PF	Polimorfismo	Sistema
CF	Acoplamento	Sistema

3 Valores de referência de métricas de software para o paradigma orientado a objetos

Neste capítulo são apresentados como foram buscados na literatura valores referência para as métricas aplicadas. Esses valores tem a finalidade de obter critérios que sirvam de base para fazer a avaliação dos softwares utilizados nesta pesquisa. Na Seção 3.1 será discutida a importância em se obter esses valores e como estes podem ajudar na gestão de projetos. Na Seção 3.2 é apresentada a busca para as métricas CK, e na Seção 3.3 é apresentada a pesquisa para as métricas MOOD. A seleção da ferramenta para a coleta de métricas é mostrada na Seção 3.4.

3.1 Valores referência para métricas de software

A melhoria da gestão depende da capacidade de identificar, medir e controlar os parâmetros essenciais do processo de desenvolvimento. Isto pode ser conseguido por meio de métricas de software para medição de tais parâmetros. As métricas de software permitem a avaliação, controle e melhoria de produtos e processos de software (FERREIRA et al., 2012), (SRINIVASAN; DEVI, 2014).

A interpretação correta das métricas é essencial para avaliar e melhorar o projeto de sistemas de software. Os valores referência têm um significado prático, teórico e metodológico. Eles são úteis para identificar classes de alto risco e mais fáceis de serem usados do que modelos estatísticos e equações que geralmente necessitam de validação (BENLARBI et al., 2000), (RIAZ; MENDES; TEMPERO, 2009).

Ao conhecer os valores referência, estes podem ser utilizados para identificação de prováveis problemas no software e, assim, melhorar a sua qualidade. Esses valores são úteis para interpretar a complexidade do projeto, pois os riscos de cada classe podem ser avaliados por meio deles (RIAZ; MENDES; TEMPERO, 2009), (SHATNAWI, 2010) (FERREIRA et al., 2012).

Apesar da importância das métricas no gerenciamento de qualidade de software orientado a objetos, este recurso não tem sido efetivamente utilizado na indústria (LIMA; RESENDE; LETHBRIDGE, 2016). Isso se deve ao fato de que os valores característicos da maioria das métricas de software ainda não são conhecidos (RIAZ; MENDES; TEMPERO, 2009), (FERREIRA et al., 2012). Desta maneira, o uso efetivo de métricas em Engenharia de Software é limitado pela

falta de conhecimento sobre os limiares métricos de software. São escassos os estudos realizados para derivar valores que sirvam como parâmetros. Sem conhecer os limites métricos, a aplicação de métricas para avaliação do software fica comprometida (MARINESCU; LANZA, 2006).

Na literatura, são encontradas pesquisas que tem como proposta sugerir valores referência para métricas específicas.

Na pesquisa relatada em (BENLARBI et al., 2000) os autores extraíram valores referência para quatro métricas do conjunto CK (WMC, DIT, NOC e RFC) avaliando dois sistemas desenvolvidos em C++. Foi utilizada regressão logística para a extração desses valores. Como resultado, os autores sugeriram um conjunto de valores referência para cada sistema avaliado, sem um consenso de valor referência para cada métrica estudada.

No trabalho retratado em (ALVES; YPMA; VISSER, 2010), os autores propuseram um método para determinar os limites métricos a partir de dados de medição. A métrica de complexidade de McCabe é utilizada como exemplo. A metodologia foi aplicada em cem sistemas de software orientados a objetos, tanto proprietários como open-source e codificados em C# e Java.

Na pesquisa apresentada em (SHATNAWI et al., 2010), os autores investigaram o uso do método ROC (ZWEIG; CAMPBELL, 1993) para identificar os limiares para prever a existência de erros em diferentes categorias. Eles realizaram uma experiência usando as métricas CK e aplicaram a técnica em três versões do Eclipse. Como resultado, os autores encontraram valores referência para algumas métricas orientadas a objetos que separavam as classes sem erro das classes que tinham erros de alto impacto.

No estudo descrito em (FERREIRA et al., 2012), os autores propuseram valores referência para seis métricas de software orientado por objetos: DIT e LCOM (conjunto CK), CF (conjunto MOOD), número de conexões aferentes, número de atributos públicos e número de métodos públicos. Esse estudo foi realizado em um conjunto de 40 softwares abertos, de diferentes contextos e tamanhos.

No trabalho apresentado em (KAUR; SINGH; KAUR, 2013), os autores utilizaram a regressão logística para investigar os valores referência das métricas CK, exceto para a métrica LCOM. Duas versões de um mesmo software foram usadas como um conjunto de dados para validar o estudo. Os autores concluíram que as métricas CK são úteis para identificar problemas estruturais no código fonte.

Como pode ser observado, as pesquisas diferem entre si em importantes aspectos, tais como metodologia, métricas abordadas, número de softwares analisados e linguagens que estes foram codificados.

Neste trabalho, os valores descritos em (JULIANO; TRAVENÇOLO; SOARES, 2014) serão utilizados como referência para o conjunto de métricas CK e são mostrados na Seção 3.2. Para as métricas MOOD, foi feita uma pesquisa para esses valores, que é apresentada na Seção

3.3.

3.2 Parâmetros para as Métricas CK

Na pesquisa descrita em (JULIANO; TRAVENÇOLO; SOARES, 2014), os autores realizaram uma revisão sistêmica sobre trabalhos que utilizaram as métricas de Chidamber e Kemerer como fonte de informação para a realização de atividades de Engenharia de Software. Como exemplo dessas atividades, pode-se citar predição a propensão a erros, impacto da refatoração nas métricas de um software, facilidade de reuso caixa-branca e análise de índices de manutenibilidade ou qualidade.

Os autores realizaram uma pesquisa estatística dos valores das métricas CK e sugeriram valores considerados como baixo, alto e anomalias, ou seja, que estão fora de um padrão de qualidade. O objetivo foi fornecer informações sobre quais métricas foram utilizadas com sucesso e sugerir valores dessas métricas que possam auxiliar atividades de Engenharia de Software.

Para cumprir os objetivos propostos, foi executada uma pesquisa por artigos entre diversas revistas científicas e anais de conferências que descrevessem os resultados na análise de software e utilizaram as métricas CK. Foram analisados softwares codificados nas linguagens Java e C++. Como resultado, foi criada uma classificação das métricas baseada em seus valores.

Com o objetivo de limitar a abrangência da pesquisa, foram selecionados estudos realizados durante os anos de 2003 a 2013. As strings utilizadas na busca foram:

- "software"AND "metrics";
- "CK"AND "metrics";
- "design"AND "defects";
- "proneness"AND "error";
- "proneness"AND "fault".

Ao considerar os títulos dos artigos, 48 estudos foram selecionados. Posteriormente, foi realizada a leitura dos resumos e os que não se adequavam à proposta do referido estudo foram descartados. Por fim, 16 artigos foram escolhidos.

Em seguida, a média e o desvio padrão de cada uma das métricas dos softwares presentes nos estudos selecionados foram tabulados. Valores discrepantes dos demais foram desconsiderados, ou seja, valores que estavam nas extremidades (25% menores e 25% maiores). Esse tipo de média é conhecido como média do intervalo interquartil (IQM - Interquartile Mean) (HUCK, 2012).

Após o cálculo da média e desvio padrão, foi criada uma classificação das métricas, baseada em um estudo proposto em (MARINESCU; LANZA, 2006). Para cada métrica CK, foram definidas quatro classes de valores: baixo, normal, alto e anomalia.

Foram classificados como baixo os valores das métricas que são menores ou iguais ao resultado da subtração do IQM da média e do desvio padrão. Os valores normais são superiores aos valores baixos e inferiores aos valores altos. Os valores das métricas que estavam acima da classificação baixo e abaixo da soma do IQM da média e IQM do desvio padrão foram considerados como alto. Por fim, são considerados como anomalia os valores das métricas que são iguais ou superiores à soma do IQM da média e o IQM do desvio padrão acrescido de 30%.

Após a classificação das métricas, os resultados dos valores das métricas para os diferentes tipos de linguagem de programação (C++ e Java) foram comparados, com o objetivo de encontrar valores diferentes de anomalias, devido às características particulares de cada uma das linguagens.

Para a realização do estudo de caso descrito no Capítulo 4, foram analisados apenas produtos de software codificados na linguagem Java. Portanto, serão considerados apenas os resultados para esta linguagem. Dos estudos que utilizaram softwares desenvolvidos em Java, oito foram selecionados. No total, foram relatados vinte softwares desenvolvidos nesta linguagem. Os trabalhos selecionados foram:

1. (SUBRAMANYAM; KRISHNAN, 2003);
2. (BENESTAD; ANDA; ARISHOLM, 2006);
3. (STROGGYLOS; SPINELLIS, 2007);
4. (SHATNAWI; LI, 2008);
5. (NAIR; SELVARANI, 2012);
6. (JOHARI; KAUR, 2012);
7. (ABUASAD; ALSMADI, 2012);
8. (KAKARONTZAS et al., 2013).

A Tabela 3.1 mostra os valores encontrados considerados como baixo, alto e anomalia para as métricas CK em produtos de software codificados em Java.

Visto que nenhuma das métricas CK possui valores negativos, os autores ressaltaram que na classificação de valores considerados como baixo, os valores abaixo de zero foram alterados para zero. Uma segunda modificação foi a aproximação dos valores considerados como anomalia, visto que as métricas CK possuem somente valores inteiros.

Tabela 3.1 – Valores referência para as métricas CK em softwares desenvolvidos na linguagem Java

	Baixo	**Normal**	**Alto**	**Anomalia**
DIT	0	1	2	3
NOC	0	1 - 5	6 - 7	8
CBO	2	3 - 14	15 - 19	20
RFC	0	1 - 64	65 - 83	84
LCOM	0	1 - 297	298 - 387	388
WMC	1	2 - 21	22 - 28	29

Fonte: (JULIANO; TRAVENÇOLO; SOARES, 2014)

3.3 Parâmetros para as Métricas MOOD

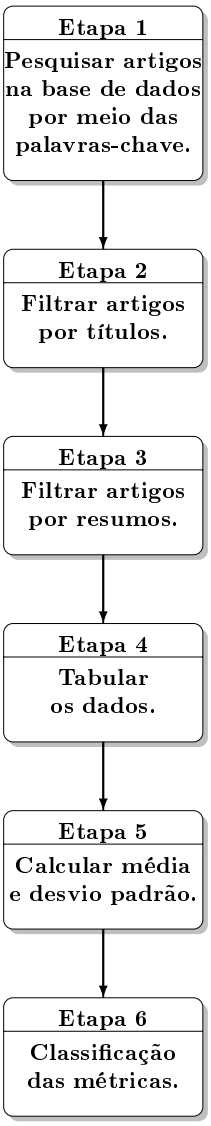
Nesta seção será apresentada uma pesquisa sobre trabalhos que utilizaram as métricas MOOD (ABREU; CARAPUÇA, 1994) como fonte de informação para a avaliação de softwares desenvolvidos no paradigma orientado a objetos.

O objetivo deste estudo foi identificar padrões de comportamento inadequados de desenvolvimento de software orientado a objetos de acordo com o conjunto de métricas MOOD. Foram buscados na literatura valores para as métricas aplicadas, com a finalidade de obter critérios que sirvam de base para fazer a comparação em relação a outros softwares desenvolvidos. Ao categorizar os valores, foram definidos quatro intervalos: baixo, normal, alto e anomalia.

Com o propósito de estabelecer o que será considerado como anomalia para as métricas do conjunto MOOD, foi realizada uma pesquisa por artigos descrevendo resultados na análise de software que utilizaram esse conjunto de métricas. Os dados foram extraídos dos artigos escolhidos, tabulados e analisados utilizando métodos estatísticos. Por fim, foi criada uma classificação das métricas baseada em seus valores.

As etapas da pesquisa são mostradas na Figura 3.1.

Figura 3.1 – Etapas do processo de seleção e análise dos estudos



Fonte: Dados do autor.

Na primeira etapa foi realizada a escolha das revistas científicas e conferências que nortearam este estudo, utilizando as seguintes base de dados:

- ACM Digital Library;
- IEEE Xplore;
- ScienceDirect;
- SpringerLink.

Para limitar a abrangência da pesquisa, somente estudos realizados durante os anos de 2006 a 2016 foram selecionados. Esse critério foi escolhido para deixar o presente estudo com

informações de artigos atuais. Devido ao baixo número de trabalhos encontrados (apenas 5), a pesquisa foi expandida por mais dez anos, englobando de 1996 a 2016.

As palavras utilizadas na pesquisa foram:

- "MOOD"AND "software metrics";
- "evaluating"AND "software".

Após a realização da pesquisa, 23 artigos foram encontrados. As segunda e terceira etapas consistiram em selecionar os trabalhos com base nos títulos e resumos, respectivamente. Os artigos que não apresentavam avaliação de software por meio do conjunto de métricas MOOD foram descartados. Dos 23 estudos, 06 foram escolhidos e são mostrados na Tabela 3.2:

Tabela 3.2 – Trabalhos selecionados

1	An Evaluation of the MOOD Set of Object-Oriented Software Metrics	(HARRISON; COUNSELL; NITHI, 1998)
2	A Complexity Metrics Set for Large-Scale Object-Oriented Software Systems	(MA et al., 2006)
3	Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes	(OLAGUE et al., 2007)
4	Object Oriented Design Metrics Framework Based on Code Extraction	(EL-LATEEF; YOUSEF; ISMAIL, 2008)
5	Empirical Comparison of Three Metrics Suites for Fault Prediction in Packages of Object-Oriented Systems: A Case Study of Eclipse	(ELISH; AL-YAFEI; AL-MULHEM, 2011)
6	Software Quality Estimation Through Object Oriented Design Metrics	(ARORA et al., 2011)

A quarta etapa consistiu em tabular os dados encontrados. No total, esses artigos relatam 23 softwares que foram analisados por meio das métricas MOOD. Nos estudos descritos em (OLAGUE et al., 2007) e (EL-LATEEF; YOUSEF; ISMAIL, 2008) não foram utilizadas todas as métricas do conjunto, entretanto estes trabalhos foram considerados devido a importância de seus resultados.

Na quinta etapa, foram calculados a média (AVG) e o desvio padrão (STDEV) dos valores encontrados. A Tabela 3.3 mostra os valores das métricas, a média e o desvio padrão obtidos dos softwares utilizados como fonte de estudo nos artigos analisados. Os valores não relatados nos artigos são representados pelo símbolo $\otimes$.

Por fim, a sexta etapa desta pesquisa consistiu em estabelecer os valores referenciais para o conjunto de métricas MOOD. Para cada métrica, foram definidas quatro classes de valores: baixo, normal, alto e anomalia. Na Tabela 3.4 é apresentado como foram feitas as classificações.

Tabela 3.3 – Valores encontrados para Métricas MOOD

Artigo	Sw	Métricas					
		MIF	AIF	MHF	AHF	PF	CF
(HARRISON; COUNSELL; NITHI, 1998)	S1	0,15	0,17	0,10	0,46	0,04	0,04
	S2	0,14	0,11	0,08	0,67	0,05	0,04
	S3	0,21	0,15	0,16	0,66	0,09	0,04
	S4	0,27	0,31	0,10	0,44	0,03	0,06
	S5	0,46	0,47	0,25	0,63	0,07	0,03
	S6	0,34	0,26	0,15	0,68	0,05	0,05
	S7	0,23	0,20	0,16	0,52	0,07	0,05
	S8	0,37	0,37	0,16	0,49	0,06	0,05
	S9	0,27	0,32	0,15	0,51	0,06	0,05
(MA et al., 2006)	S10	0,34	0,11	0,18	0,62	0,05	0,10
(OLAGUE et al., 2007)	S11	0,21	0,24	0,68	0,07	⊗	⊗
	S12	0,20	0,24	0,64	0,06	⊗	⊗
	S13	0,17	0,17	0,62	0,05	⊗	⊗
	S14	0,17	0,17	0,61	0,05	⊗	⊗
	S15	0,17	0,17	0,57	0,05	⊗	⊗
	S16	0,20	0,18	0,54	0,06	⊗	⊗
(EL-LATEEF; YOUSEF; ISMAIL, 2008)	S17	0,14	⊗	⊗	⊗	0,07	⊗
	S18	0,12	⊗	⊗	⊗	0,07	⊗
	S19	0,11	⊗	⊗	⊗	0,07	⊗
	S20	0,11	⊗	⊗	⊗	0,07	⊗
	S21	0,21	⊗	⊗	⊗	0,07	⊗
(ELISH; AL-YAFEI; AL-MULHEM, 2011)	S22	0,18	0,19	0,21	0,10	0,25	0,08
(ARORA et al., 2011)	S23	0,02	0,06	0,89	0,95	0,10	⊗
AVG		0,20	0,22	0,35	0,39	0,07	0,05
STDEV		0,10	0,10	0,26	0,29	0,05	0,02

Tabela 3.4 – Proposta de classificação das métricas

Classificação	**Valor**
Baixo	Baixo ≤AVG - STDEV
Normal	Baixo <Normal <Alto
Alto	AVG + STDEV ≤Alto ≤ Anomalia
Anomalia	Anomalia ≥ AVG + STDEV + 50%*STDEV

São classificados como baixo os valores das métricas que são menores ou iguais ao resultado da subtração entre média e o desvio padrão. Os valores normais estão entre os valores baixo e alto. São considerados como alto os valores das métricas que são maiores ou iguais ao resultado da soma entre média e o desvio padrão e abaixo da anomalia. Por fim, uma anomalia é classificada de acordo com a proposta apresentada (MARINESCU; LANZA, 2006), sendo calculada pela soma entre média e desvio padrão acrescido de 50%.

Utilizando os valores da Tabela 3.3, outra tabela foi elaborada, com as classificações das métricas. Na Tabela 3.5 são mostrados os valores das métricas já classificados por meio das equações expostas na Tabela 3.4.

Tabela 3.5 – Valores referência para Métricas MOOD

Métricas	**Valores**			
	Baixo	**Normal**	**Alto**	**Anomalia**
MIF	≤ 0,10	0,11 - 0,29	0,30 - 0,34	≥ 0,35
AIF	≤ 0,12	0,13 - 0,31	0,32 - 0,36	≥ 0,37
MHF	≤ 0,09	0,10 - 0,60	0,61 - 0,73	≥ 0,74
AHF	≤ 0,10	0,11 - 0,67	0,68 - 0,82	≥ 0,83
PF	≤ 0,02	0,03 - 0,11	0,12 - 0,13	≥ 0,14
CF	≤ 0,03	0,04 - 0,06	0,07	≥ 0,08

Os valores apresentados na Tabela 3.5 serão utilizados como parâmetros nesta pesquisa para avaliação dos softwares durante a realização do estudo de caso.

3.4 Ferramenta de coleta das métricas

Para a coleta das métricas, foi realizada uma pesquisa com o objetivo de encontrar uma ferramenta que realizasse essa tarefa de maneira automática e obedecesse aos critérios de gratuidade, abranger os dois conjuntos de métricas envolvidos na pesquisa e o resultado ser mostrado de maneira gráfica.

Após essa busca, quatro ferramentas foram selecionadas para serem analisadas conforme os critérios estabelecidos :

1. CKJM - Chidamber and Kemerer Java Metrics;
2. CCCC - C and C++ Code Counter;
3. JHawk 5;
4. IntelliJ Idea.

As características de cada ferramenta são explanadas a seguir:

1. **CKJM - Chidamber and Kemerer Java Metrics**

CKJM (SPINELLIS, 2005) é uma ferramenta livre que faz a análise de código Java por meio das métricas CK.

O software analisa os arquivos com extensão *.class* contidos em uma pasta e retorna os resultados na sequência WMC, DIT, NOC, CBO, RFC, LCOM, Acoplamento Aferente (MARTIN, 1994) e Número de Métodos Públicos (BANSIYA; DAVIS, 2002) .

Na Figura 3.2 é mostrada a saída de tela desta ferramenta com os resultados coletados em um código teste.

```
C:\Users\Jamille\Desktop\metricas\ckjm-1.9\ckjm-1.9\build>java -jar ckjm-1.9.jar
 C:\Users\Jamille\Desktop\metricas\Heranca\bin\*class
Error obtaining all superclasses of public class TV extends Eletrodomestico
filename                C:\Users\Jamille\Desktop\metricas\Heranca\bin\TV.class
compiled from           TV.java
compiler version        52.0
access flags            33
constant pool           35 entries
ACC_SUPER flag          true

Attribute(s):
        SourceFile(TV.java)

3 fields:
        private int canal
        private int volume
        private int tamanho

7 methods:
        public void <init>(int, int, int, int, int)
        public int getCanal()
        public void setCanal(int)
        public int getVolume()
        public void setVolume(int)
        public int getTamanho()
        public void setTamanho(int)

Eletrodomestico 7 1 1 0 8 21 1 7
ExemploHeranca 3 1 0 1 4 3 0 3
TV 7 0 0 1 8 21 1 7
```

Figura 3.2 – Saída de tela da ferramenta CKJM

2. **CCCC - C and C++ Code Counter**

CCCC (LITTLEFAIR; WATKINS; MARCUS, 2013) é um software livre que faz a análise de código Java e C++ utilizando um conjunto de métricas. Considerando as métricas CK, as utilizadas são WMC, CBO, NOC e DIT.

Para aplicar as métricas em códigos C++, é preciso informar arquivos com a extensão *.cpp*, enquanto que para códigos Java, os arquivos analisados devem possuir a extensão *.java*. Como saída, é criado um arquivo *html* com os resultados de todas as métricas.

Na Figura 3.3 é apresentada a saída de tela com os resultados coletados em um código experimental, com destaque para as métricas CK.

Metric	Tag	Overall	Per Function
Lines of Code	LOC	7	******
McCabe's Cyclomatic Number	MVG	1	******
Lines of Comment	COM	6	********
LOC/COM	L_C	------	
MVG/COM	M_C	------	
Weighted Methods per Class (weighting = unity)	WMC1	2	
Weighted Methods per Class (weighting = visible)	WMCv	0	
Depth of Inheritance Tree	DIT	0	
Number of Children	NOC	0	
Coupling between objects	CBO	1	
Information Flow measure (inclusive)	IF4	0	********
Information Flow measure (visible)	IF4v	0	********
Information Flow measure (concrete)	IF4c	0	********

Figura 3.3 – Saída de tela da ferramenta CCCC

3. **JHawk 5**

JHawk 5 (HALL, 2016) é uma ferramenta proprietária, porém oferece versão experimental por 15 dias. A ferramenta realiza a análise de código Java por meio de várias métricas, incluindo as do conjunto CK.

Na Figura 3.4 é exibida a saída de tela desta ferramenta com os resultados coletados em um código teste. Os resultados para as métricas CK foram realçados.

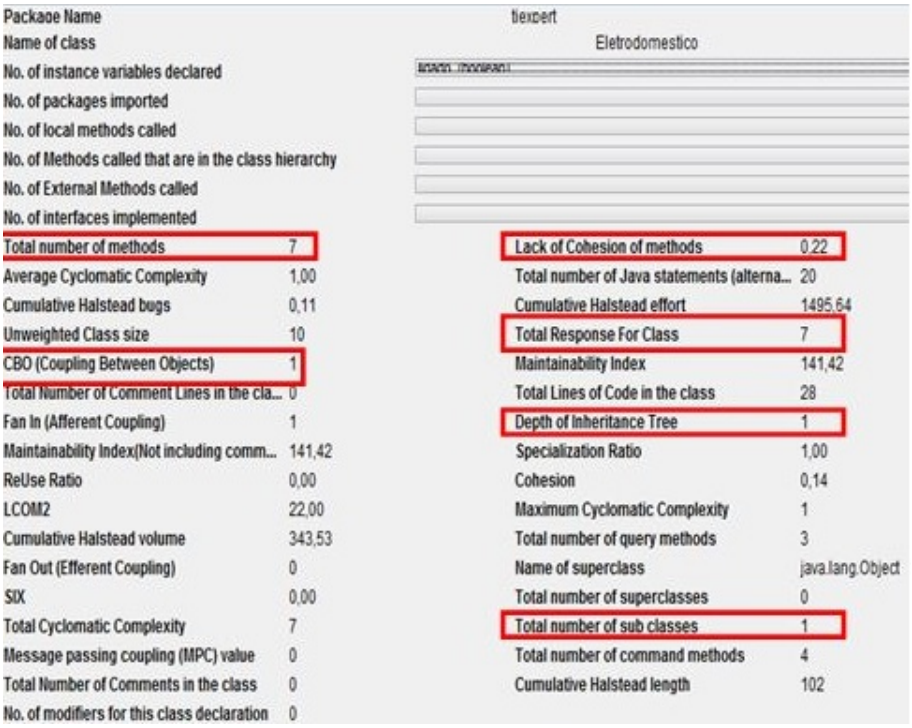
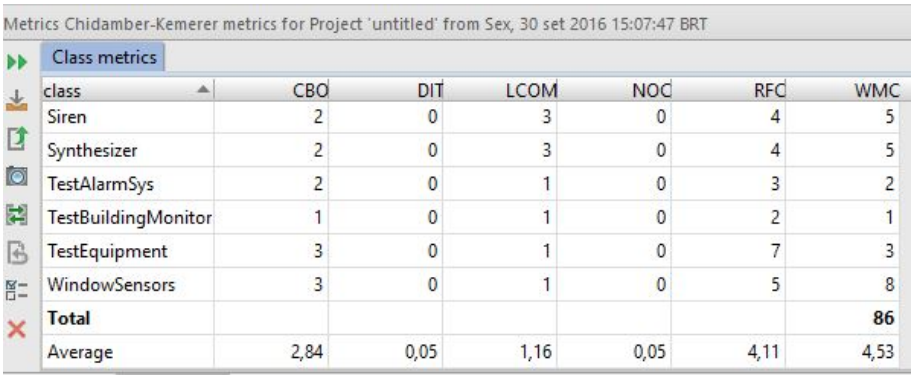


Figura 3.4 – Saída de tela da ferramenta JHawk

4. **IntelliJ Idea**

IntelliJ IDEA (BRAINS J, 2016) é uma plataforma sobre a qual parte das funcionalidades é fornecida na forma de plugins. É uma ferramenta proprietária, mas existe uma versão livre. Por meio dela, é possível calcular tanto as métricas do conjunto CK quanto MOOD, além de outras.

Nas Figuras 3.5 e 3.6 são mostrados exemplos de resultados das métricas CK e MOOD, respectivamente.



class	CBO	DIT	LCOM	NOC	RFC	WMC
Siren	2	0	3	0	4	5
Synthesizer	2	0	3	0	4	5
TestAlarmSys	2	0	1	0	3	2
TestBuildingMonitor	1	0	1	0	2	1
TestEquipment	3	0	1	0	7	3
WindowSensors	3	0	1	0	5	8
Total						**86**
Average	2,84	0,05	1,16	0,05	4,11	4,53

Figura 3.5 – Resultado das métricas CK

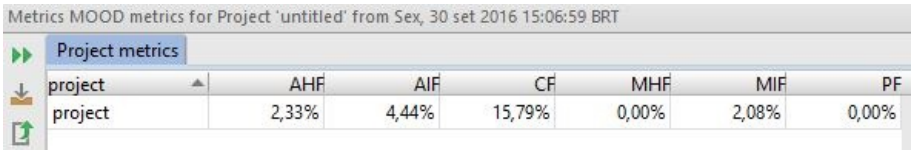


Figura 3.6 – Resultado das métricas MOOD

É possível também exportar os resultados para um arquivo de extensão *.csv*, facilitando a tabulação dos dados, principalmente se o software analisado contiver muitas classes. Na Figura 3.7 é mostrado um exemplo desse tipo de saída.

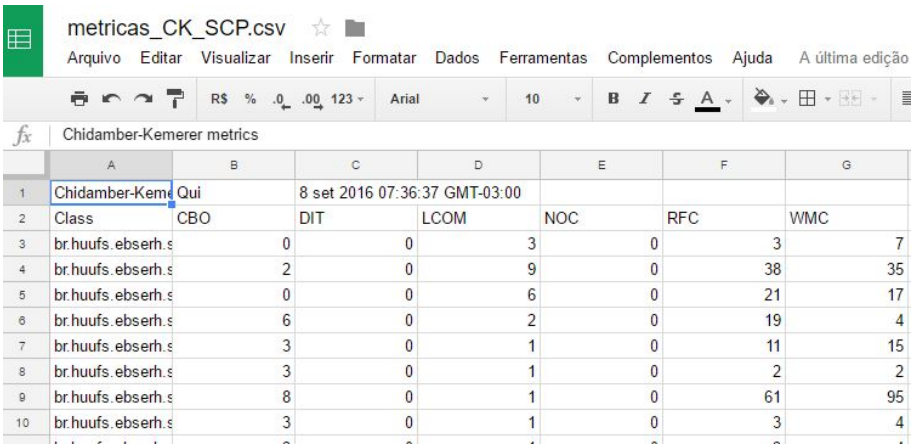


Figura 3.7 – Resultado das métricas em arquivo *.csv*

Na Tabela 3.6 são mostrados os critérios usados para a avaliação e escolha das ferramentas.

Tabela 3.6 – Critérios de escolha das ferramentas

Ferramenta	**Critérios**			
	Gratuita	**Usabilidade**	**Métricas CK**	**Métricas MOOD**
CKJM	Sim	Difícil tabulação para códigos grandes	Sim	Não
CCCC	Sim	Difícil tabulação para códigos grandes	Não todas do conjunto	Não
JHawk	Paga, mas com versão experimental	Difícil tabulação para códigos grandes	Sim	Não
IntelliJ Idea	Paga, mas com versão livre	Dados já vem tabulados	Sim	Sim

Após a análise dos critérios, a ferramenta selecionada foi a IntelliJ Idea. Esta é a única ferramenta que calcula todas as métricas dos dois conjuntos utilizados nessa pesquisa, além de exportar os resultados em arquivo *.csv*.

4 Estudo de caso

Neste capítulo é apresentada a execução do estudo de caso realizado em empresas e universidades locais. Na Seção 4.1 será abordado o planejamento, contendo todas as suas etapas. Na Seção 4.2 é relatada a operação do experimento. Por fim, na Seção 4.3 os resultados são apresentados.

4.1 Planejamento

Um estudo de caso pode ser definido como a investigação de um fenômeno dentro do seu contexto real, especialmente quando não há uma distinção clara entre o fenômeno e seu contexto (WOHLIN et al., 2012). Pode ser classificado como exploratório ou confirmatório. O primeiro visa derivar novas hipóteses e construir teorias, e o segundo tem como objetivo confirmar teorias existentes (PERRY; SIM; EASTERBROOK, 2004). Este experimento caracteriza-se como exploratório,sendo que as hipóteses são apresentadas na Seção 4.1.2.

Nesta seção serão retratados a definição e o planejamento do experimento, seguindo as diretrizes apresentadas em (WOHLIN et al., 2012).

4.1.1 Definição do objetivo

O objetivo deste trabalho foi utilizar métricas para avaliar produtos de software desenvolvidos de acordo com o paradigma orientado a objeto. Os resultados foram utilizados para avaliar o desempenho da equipe de desenvolvimento.

O experimento foi realizado com programadores dos setores de desenvolvimento de duas empresas públicas e estudantes de duas universidades locais utilizando softwares codificados por meio do paradigma orientado a objetos.

O objetivo do experimento foi formalizado utilizando o modelo GQM proposto originalmente por Basili e Weiss, no trabalho descrito em (BASILI; WEISS, 1984):

- **Analisar** a aplicabilidade de métricas para avaliar a qualidade do produto de software;
- **Com a finalidade** de avaliar tecnicamente os membros da equipe de desenvolvimento de software;
- **Com respeito** à qualidade do software produzido individualmente ou em duplas;
- **Do ponto de vista** de programadores e gestores de projetos de desenvolvimento de software;

- **No contexto** de programadores dos setores de desenvolvimento de duas empresas públicas locais e alunos de graduação.

4.1.2 Formulação das suposições

As questões a serem respondidas nessa pesquisa foram:

1. Métricas de software orientadas a objetos podem ser usadas para avaliar a qualidade do produto final do ponto de vista do gerenciamento de projetos?
2. É possível avaliar tecnicamente equipes de desenvolvimento de software por meio de métricas de produto de software?

Para responder a essas questões, foram utilizadas métricas da Engenharia de Software que têm sido aplicadas na medição do produto de software. As métricas escolhidas foram descritas no Capítulo 2, Seção 2.2.

Após definição dos objetivos e métricas, foram consideradas as seguintes suposições:

- Suposição 1:

  SP1: Métricas de produto de software são úteis no gerenciamento de projetos.

- Suposição 2:

  SP2: Equipes de desenvolvimento de software podem ser avaliadas tecnicamente por meio de métricas de produto de software.

4.1.3 Variável independente

As variáveis independentes deste experimento foram os softwares avaliados, a aplicação de métricas orientadas a objeto e a ferramenta para coleta das métricas. As métricas aplicadas foram apresentadas no Capítulo 2, Seção 2.2 e a seleção da ferramenta foi descrita no Capítulo 3, Seção 3.4.

4.1.4 Variáveis dependentes

As variáveis dependentes abordadas no experimento foram:

- **Média dos valores das métricas:** para cada métrica, foi calculada a média por meio da razão entre a soma de todos os valores e a quantidade total de valores.
- **Maior dos valores das métricas:** para cada métrica, foi observado qual o maior entre todos os valores.

- **Menor dos valores das métricas:** para cada métrica, foi observado qual o menor entre todos os valores.

4.1.5 Variável interveniente

A variável que poderia influenciar no experimento foi a quantidade de participantes do projeto. Adotando os critérios descritos na Seção 4.1.6, apenas seis desenvolvedores foram selecionados para participar nas empresas em que o experimento foi realizado. Nas universidades, somente nove alunos concluíram a atividade proposta. No total, quinze pessoas participaram do estudo. Este número pode ser considerado baixo para a realização desta pesquisa.

4.1.6 Seleção de participantes

O experimento foi realizado em dois grupos distintos:

- **Programadores em seu ambiente de trabalho:** duas empresas públicas propiciaram que o estudo de caso fosse aplicado em suas respectivas equipes de desenvolvimento de software. Cada uma concedeu três desenvolvedores para a realização da pesquisa.

  Para garantir a eficácia do experimento, foi solicitado que os produtos de software analisados tivessem sido codificados por apenas um programador, ou no máximo por uma dupla.

  Para a realização do estudo de caso, não haveria tempo para analisar o projeto desde a sua concepção até a sua finalização. Por isso, foi necessário avaliar softwares já concluídos, para oportunizar que a aplicação das métricas ocorressem em tempo hábil para a pesquisa. Desta maneira, para que pudesse ser feita de forma mais direta a avaliação do desenvolvedor por meio do software desenvolvido, foi solicitado às empresas que os softwares tivessem sido codificados por no máximo dois indivíduos.

- **Alunos de graduação:** duas universidades oportunizaram a participação de seus alunos nesta pesquisa.

  Com o propósito de garantir a eficácia do experimento, em cada universidade foi solicitado pelo professor que todos os alunos desenvolvessem um projeto para resolver um único problema. Para selecionar os participantes, foram escolhidos aqueles cujos projetos resolvessem o problema proposto com completude.

4.1.7 Projeto do experimento

O experimento foi projetado para avaliar a equipe de desenvolvimento por meio da qualidade do software produzido.

Para execução do experimento, duas atividades foram realizadas:

- Questionário on line para coletar dados da equipe, sendo este aplicado apenas nas empresas. O questionário é apresentado no Apêndice A;
- Coleta de métricas de software orientadas a objetos.

4.1.8 Instrumentação

As ferramentas usadas no experimento foram:

- **Questionário:** questionário aplicado aos desenvolvedores nas empresas, com o objetivo de obter informações sobre escolaridade e experiências dos participantes;
- **IntelliJ Idea:** coleta automaticamente as métricas de software e exibe os resultados em uma tabela.

4.2 Operação do experimento

Para realização do experimento, primeiro foi aplicada a preparação e posteriormente a execução, descritas nas Seções 4.2.1 e 4.2.2.

4.2.1 Preparação

Para preparar o experimento, foram seguidas as seguintes etapas:

- **Questionário:** foi definido que os participantes responderiam um questionário acerca de sua escolaridade, experiência na área de tecnologia da informação, em programação, tempo de trabalho na empresa, dentre outras informações;
- **Treinamento da ferramenta IntelliJ Idea:** foi feito um treinamento com os programadores de aproximadamente 30 minutos sobre a ferramenta, desde a instalação até a maneira de calcular as métricas. As métricas dos softwares desenvolvidos pelos alunos foram coletadas pelos respectivos professores, que também receberam treinamento.

4.2.2 Execução

Ao final das etapas anteriores, foi iniciado o experimento, seguido de acordo com o planejamento descrito na Subseção 4.1.7.

Coleta dos dados

Dadas as instruções, todos os participantes afirmaram não ter dúvidas sobre a condução do experimento. Em seguida, concordaram em responder o questionário e coletar as métricas propostas.

Para os softwares selecionados a participar do experimento, foram coletadas todas as métricas dos conjuntos CK e MOOD, explanadas no Capítulo 2. Os resultados foram comparados com os valores descritos no Capítulo 3.

4.3 Resultados

Nesta seção são apresentados os resultados da pesquisa. Inicialmente são mostradas as informações coletadas acerca dos programadores das empresas participantes e dos alunos de graduação. Em seguida, os resultados das métricas coletadas são expostos.

4.3.1 Dados coletados nas empresas participantes

Para que as questões da pesquisa pudessem ser respondidas, foram analisadas a escolaridade e experiência dos membros da equipe e a qualidade do software produzido.

A pesquisa foi realizada no setor de desenvolvimento de duas empresas públicas. Na Tabela 4.1 é mostrada a relação entre as empresas, os programadores e seus respectivos produtos de software.

Tabela 4.1 – Empresa x Software x Programador

Empresa	Software	Programador
E1	S1	P1, P2
	S2	P3
E2	S3	P4, P5
	S4	P6

Na Tabela 4.2 é mostrada a idade de cada programador, assim como o nível de escolaridade e se possui alguma certificação na área. Na Tabela 4.3 é apresentada a experiência profissional na área de Tecnologia da Informação, não necessariamente em programação. Na Tabela 4.4 é exposta a experiência profissional em programação. Na Tabela 4.5 é mostrado o tempo de experiência na linguagem Java de cada programador.

Tabela 4.2 – Escolaridade

Programador	**Idade**	**Escolaridade**	**Certificação**	**Capacitação em andamento**
P1	25	Graduação		Mestrado
P2	27	Graduação	X	
P3	28	Graduação		
P4	26	Graduação		
P5	38	Graduação		
P6	30	Graduação	X	Mestrado

Tabela 4.3 – Experiência em Tecnologia da Informação

Tempo	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
Até 1 ano						
Entre 2 e 4 anos	X			X		
Entre 5 e 7 anos		X	X			
Entre 8 e 10 anos					X	X
Mais de 10 anos						

Tabela 4.4 – Experiência em Programação

Tempo	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
Até 1 ano						
Entre 2 e 4 anos	X			X		
Entre 5 e 7 anos		X	X			
Entre 8 e 10 anos					X	X
Mais de 10 anos						

Tabela 4.5 – Experiência na Linguagem Java

Programador	**Tempo**
P1	4 anos
P2	7 anos
P3	5 anos
P4	4 anos
P5	10 anos
P6	2 anos

Também foi perguntado aos programadores há quanto tempo eles trabalham na empresa, e foi constatado que apenas um participante (P6) trabalha entre 8 e 10 anos. Os demais afirmaram o período entre 2 e 4 anos.

Na Tabela 4.6 são mostrados os valores encontrados para as métricas CK coletados nos softwares desenvolvidos nas empresas, onde as siglas significam: SW = Software, NC = número de classes total, ME = método estatístico de avaliação, AVG = média, MAX = valor máximo e, por fim, MIN = valor mínimo. As siglas referentes às métricas CK foram explanadas na Seção 2.2.1 do Capítulo 2.

Tabela 4.6 – Valores encontrados para as métricas CK nas empresas

SW	**NC**	**ME**	**CK Metrics**					
			DIT	**NOC**	**CBO**	**RFC**	**LCOM**	**WMC**
S1	18	Avg	1,00	0,38	3,11	19,72	3,06	19,56
		Max	1,00	1,00	8,00	48,00	8,00	46,00
		Min	1,00	0,00	0,00	1,00	1,00	1,00
S2	6	Avg	1,00	0,33	1,17	56,17	3,00	79,00
		Max	1,00	1,00	4,00	137,00	7,00	176,00
		Min	1,00	0,00	0,00	2,00	1,00	3,00
S3	1053	Avg	1,00	0,57	7,52	14,43	2,40	13,59
		Max	4,00	172,00	296,00	132,00	51,00	183,00
		Min	0,00	0,00	0,00	0,00	0,00	0,00
S4	40	Avg	0,00	0,00	2,37	11,40	2,57	12,23
		Max	0,00	0,00	8,00	61,00	18,00	95,00
		Min	0,00	0,00	0,00	0,00	0,00	0,00

Na Tabela 4.7 são apresentados os valores encontrados para as métricas MOOD coletados nos softwares desenvolvidos nas empresas, onde as siglas significam: SW = Software, NC = número de classes total. As siglas referentes às métricas MOOD foram descritas na Seção 2.2.2 do Capítulo 2.

Tabela 4.7 – Valores encontrados para as métricas MOOD nas empresas

SW	**NC**	**MOOD Metrics**					
		MIF	**AIF**	**MHF**	**AHF**	**PF**	**CF**
S1	18	0,00	0,00	0,16	0,90	1,00	0,18
S2	6	0,00	0,00	0,29	0,89	1,00	0,50
S3	1053	0,65	0,50	0,09	0,97	0,02	0,01
S4	40	0,00	0,00	0,02	0,88	1,00	0,07

4.3.2 Dados coletados sobre os alunos de graduação

No total, participaram nove alunos de graduação em duas universidades diferentes. Todos os cursos possuem dez períodos de duração. As informações são mostradas na Tabela 4.8.

Tabela 4.8 – Informações sobre alunos

Universidade	**Faculdade**	**Estudante**	**Período**	**Idade**
U1	Sistemas de Informação	ST1	9	24
		ST2	4	23
		ST3	9	23
		ST4	9	23
	Ciência da Computação	ST5	9	22
U2	Sistemas de Informação	ST6	9	24
		ST7	9	23
		ST8	8	23
		ST9	6	23

Na Tabela 4.9 são expostos os valores encontrados para as métricas CK coletados nos softwares desenvolvidos pelos estudantes. As siglas têm o mesmo significado da Tabela 4.6, exceto para a primeira coluna, onde ST = estudante.

Tabela 4.9 – Valores encontrados para as métricas CK nos softwares desenvolvidos pelos estudantes

ST	**NC**	**ME**	**Métricas CK**					
			DIT	**NOC**	**CBO**	**RFC**	**LCOM**	**WMC**
ST1	04	Avg	2,00	1,33	2,00	67,50	0,75	67,00
		Max	6,00	1,00	4,00	148,00	1,00	190,00
		Min	0,00	0,00	1,00	0,00	0,00	0,00
ST2	06	Avg	1,00	1,00	3,66	10,00	1,83	6,16
		Max	1,00	1,00	5,00	25,00	5,00	13,00
		Min	1,00	1,00	2,00	2,00	1,00	2,00
ST3	07	Avg	1,00	1,00	1,71	7,29	1,86	9,43
		Max	1,00	1,00	3,00	20,00	3,00	22,00
		Min	1,00	1,00	0,00	1,00	1,00	1,00
ST4	04	Avg	1,00	1,00	3,00	22,5	3,25	57,50
		Max	1,00	1,00	3,00	26,00	6,00	95,00
		Min	1,00	1,00	3,00	19,00	1,00	36,00
ST5	25	Avg	1,13	0,13	1,33	7,58	1,96	8,08
		Max	2,00	3,00	3,00	38,00	5,00	38,00
		Min	1,00	0,00	0,00	0,00	0,00	0,00
ST6	11	Avg	1,91	0,18	2,91	10,91	2,45	10,18
		Max	3,00	2,00	9,00	42,00	5,00	40,00
		Min	1,00	0,00	1,00	2,00	0,00	1,00
ST7	12	Avg	2,31	0,23	2,46	7,62	1,62	5,23
		Max	3,00	2,00	10,00	25,00	6,00	18,00
		Min	1,00	0,00	1,00	3,00	0,00	2,00
ST8	05	Avg	1,60	1,20	3,20	9,80	1,80	6,60
		Max	2,00	2,00	4,00	12,00	5,00	9,00
		Min	1,00	0,00	3,00	6,00	1,00	1,00
ST9	05	Avg	1,80	0,60	2,40	10,80	2,20	7,80
		Max	3,00	2,00	3,00	15,00	4,00	12,00
		Min	1,00	0,00	2,00	6,00	1,00	1,00

Na Tabela 4.10 são apresentados os valores encontrados para as métricas MOOD coletados nos softwares desenvolvidos pelos estudantes. As siglas têm o mesmo significado da Tabela 4.7, exceto para a primeira coluna, onde ST = estudante.

Tabela 4.10 – Valores encontrados para as métricas MOOD nos softwares desenvolvidos pelos estudantes

ST	**NC**	**Métricas MOOD**					
		MIF	**AIF**	**MHF**	**AHF**	**PF**	**CF**
ST1	04	0,00	0,71	0,66	0,93	1,00	1,00
ST2	06	0,00	0,00	0,15	0,90	1,00	0,87
ST3	07	0,00	0,26	0,63	0,73	1,00	0,50
ST4	04	0,00	0,00	0,14	1,00	1,00	1,00
ST5	25	0,00	0,00	0,35	0,78	1,00	0,09
ST6	11	0,13	0,13	0,00	1,00	0,14	0,29
ST7	12	0,24	0,28	0,00	0,94	0,24	0,21
ST8	05	0,80	0,78	0,00	0,00	0,02	0,80
ST9	05	0,48	0,35	0,00	0,82	0,06	0,60

4.4 Ameaças à validade

Embora os resultados do experimento tenham se mostrado satisfatórios, o mesmo apresenta ameaças à sua validade que não podem ser desconsideradas.

Ameaças à validade interna: os softwares selecionados foram escolhidos seguindo o critério de ter sido codificado no máximo em dupla, mas existiu a dúvida de que outros programadores tenham participado do desenvolvimento. Esta ameaça foi mitigada com o comprometimento dos chefes dos setores e professores de que as informações passadas foram verdadeiras.

Ameaças à validade externa: o baixo número de participantes pode ser uma ameaça, visto que pode influenciar negativamente os resultados do experimento. Esta ameaça foi reduzida realizando o experimento em duas empresas e duas universidades diferentes, o que significa que a cultura ambiental não influenciou o experimento.

Ameaças à validade de construção:

1. **O experimento nas empresas ter sido executado apenas em instituições públicas:**

   Nesta pesquisa, apenas um tipo de ambiente foi estudado. Em empresas privadas, a metodologia aplicada na gestão de projetos poderia revelar resultados diferentes.

2. **O experimento ter sido executado com alunos de graduação de diferentes cursos e semestres:**

   O grau de aprendizado pode influenciar no resultado. Na pesquisa, participaram alunos que ainda estão na metade curso e outros que já estão na fase final.

5 Avaliação das equipes de desenvolvimento

Neste capítulo, é apresentada a avaliação de equipes de desenvolvimento por meio de métricas de software. Na Seção 5.1 será explanado como as métricas do paradigma orientado a objetos podem contribuir em atividades da Engenharia de Software. Nas Subseções 5.1.1 e 5.1.2 é abordado como as métricas CK e MOOD podem colaborar nessas atividades, nesta ordem. Por fim, na Seção 5.2 serão discutidos os resultados do estudo de caso, baseando-se na discussão da Seção 5.1 com os valores descritos no Capítulo 3.

5.1 Uso de Métricas orientadas a objetos em atividades de Engenharia de Software

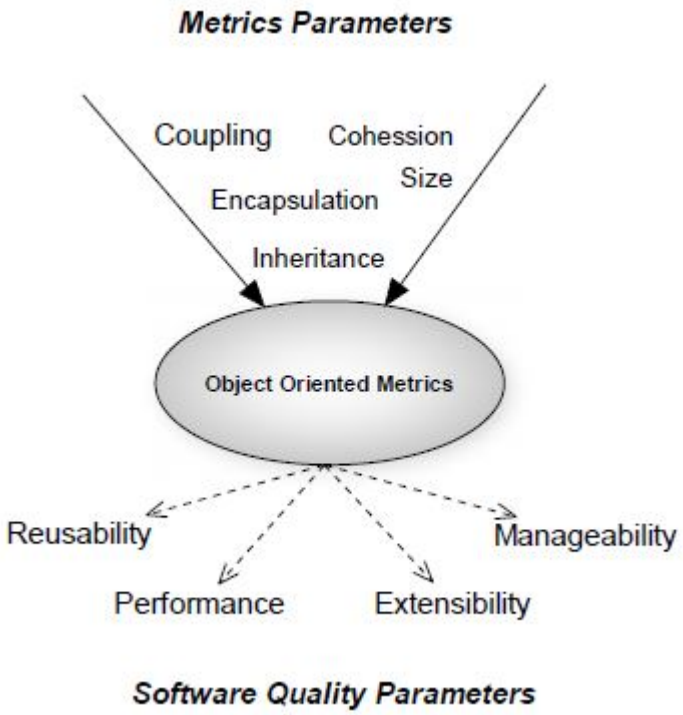
O desenvolvimento de sistemas de software grandes e complexos é um desafio. Atividades de apoio ao gerenciamento de projetos de software são uma área importante de pesquisa. A predição de defeitos é uma questão essencial na Engenharia de Software. Pode ser usada tanto na avaliação da qualidade do produto final quanto no gerenciamento de decisões durante o projeto (MOSER; PEDRYCZ; SUCCI, 2008), (LESSMANN et al., 2008). O trabalho atual de predição de defeitos concentra-se em estimar o número de defeitos remanescentes em sistemas de software, descobrir associações entre os defeitos e classificar a vulnerabilidade dos componentes de software (SONG et al., 2011).

Diversas pesquisas (MENZIES; GREENWALD; FRANK, 2007), (SINGH; KAUR; MALHOTRA, 2010), (SHATNAWI, 2010), (JOHARI; KAUR, 2012), (HUDA; ARYA; KHAN, 2015) foram feitas com o objetivo de avaliar a relação entre defeitos de software e métricas de código do paradigma orientado a objetos. Características de software como manutenibilidade, testabilidade, compreensibilidade, reutilização e eficiência podem ser medidas usando métricas de software (ZHOU; LEUNG, 2006), (OLBRICH et al., 2009), (GANDHI; BHATIA, 2010). Desta maneira, pode-se afirmar que a utilização e análise de métricas durante o processo de desenvolvimento de softwares pode melhorar a qualidade dos produtos.

A aplicação das métricas deve ser utilizada o mais cedo possível para nortear o desenvolvimento e reduzir tempo e custo dos projetos. Essas medições permitem aos projetistas acessar o software no início do processo, fazendo alterações que reduzam a complexidade e melhoram a capacidade contínua do projeto. (SHATNAWI; LI, 2008), (JASSIM; ALTAANI, 2013).

É importante ressaltar que para conhecer melhor a estrutura interna do produto de software, deve-se saber mais sobre as relações existentes entre os parâmetros de métricas e os de qualidade de software. A Figura 5.1 mostra essa relação (ARORA et al., 2011).

Figura 5.1 – Métricas vs Qualidade de software



Fonte: (ARORA et al., 2011)

Nas Seções 5.1.1 e 5.1.2 será abordado como as métricas utilizadas nesta pesquisa podem ajudar a avaliar e melhorar a qualidade do software produzido.

5.1.1 Uso das Métricas CK em atividades de Engenharia de Software

No trabalho apresentado em (BENESTAD; ANDA; ARISHOLM, 2006), foi feita uma análise do impacto de métricas estruturais, de acoplamento e coesão na avaliação da manutenibilidade de software. Os autores concluíram que a análise das métricas, quando combinadas com outras análises estatísticas, pode ser mais eficiente do que os métodos separados. Valores altos para as métricas WMC, DIT e NOC indicam maior dificuldade na realização de manutenções no software.

No trabalho descrito em (ZHOU; LEUNG, 2006), os autores utilizaram regressão logística e métodos de aprendizado de máquina para investigar a utilidade de métricas de projeto orientadas a objetos. As métricas do conjunto CK foram avaliadas na previsão de falhas quando se leva em consideração a gravidade destas falhas. Como resultado, observou-se que as métricas CBO, WMC, RFC e LCOM são estatisticamente significativas na gravidade da falha, enquanto a

DIT não é significativa para qualquer gravidade de falha. NOC é estatisticamente significante em relação a falhas de baixa gravidade.

Na pesquisa apresentada em (GILL; SIKKA, 2011), são estudadas métricas que avaliam a reutilização de softwares desenvolvidos por meio do paradigma orientado a objetos. Os autores avaliaram métricas dos conjuntos CK e MOOD e propuseram cinco novas métricas. Do conjunto CK, foram analisadas as métricas DIT e NOC e do conjunto MOOD, as métricas MIF e AIF.

No estudo descrito em (NAIR; SELVARANI, 2012) é discutida a propensão a erros em projetos de software e o seu impacto na qualidade do produto. Na realização da pesquisa foram utilizados cinco projetos comerciais e todas as métricas do conjunto CK foram analisadas. Os autores concluíram que as métricas NOC, DIT e CBO são eficazes em avaliar a propensão a erros.

Na pesquisa relatada em (JOHARI; KAUR, 2012) foi analisada a aplicabilidade das métricas CK na estimativa do esforço de manutenção de software. O estudo foi realizado utilizando softwares de código aberto com o objetivo de verificar a aplicabilidade das métricas em estimar o esforço necessário para a realização de revisões nas classes de um sistema. Os autores concluíram que as métricas WMC, RFC e CBO foram eficientes ao predizer a propensão a erros em classes.

Na pesquisa relatada em (SRIVASTAVA; KUMAR, 2013), os autores afirmam que a medição desempenha um papel importante em todas as fases do processo de desenvolvimento de software. O trabalho enfatiza a medição de atributos de qualidade diferentes, tais como a capacidade de reutilização, facilidade de manutenção, capacidade de teste, confiabilidade e eficiência. As métricas CK foram utilizadas para a medição destes atributos. Foi analisado que o alto acoplamento entre objetos e baixa coesão dos valores dos métodos tem efeito negativo sobre todos os atributos de qualidade. Profundidade da árvore de herança e número de filhos tende a aumentar a capacidade de reutilização e eficiência, mas prejudicam a capacidade de realização de testes e manutenção. Os autores concluíram que para desenvolver software de alta qualidade, o acoplamento entre os objetos e a falta de coesão de métodos devem ser mantidos baixos. Um nível moderado de profundidade da árvore de herança e número de filhos são úteis na construção de software de alta qualidade.

Na Tabela 5.1 são sumarizados os parâmetros de avaliação que podem ser utilizados na análise da qualidade de um software. Esses parâmetros são baseados nos trabalhos supracitados.

Tabela 5.1 – Uso de Métricas CK em Engenharia de Software

Parâmetros	**Métricas CK**					
	DIT	**NOC**	**CBO**	**RFC**	**LCOM**	**WMC**
Capacidade de teste	X	X	X	X		
Eficiência	X	X				
Reutilização	X	X	X			X
Manutenção	X	X	X	X		X
Falha no projeto das classes					X	
Tempo de desenvolvimento						X
Propensão a erros	X	X	X	X	X	X

5.1.2 Uso das Métricas MOOD em atividades de Engenharia de Software

O trabalho apresentado em (GUPTA; BATRA et al., 2012) traz uma discussão sobre os conjuntos de métricas CK, MOOD e Li (LI; HENRY, 1993). Os pontos fortes e fracos de todos os conjuntos são identificados. Os autores concluíram que nenhum conjunto é completo e não existe uma única métrica que possa medir todos os aspectos de um sistema orientado a objeto. Do conjunto MOOD, as métricas MIF e AIF são úteis para avaliar a capacidade de teste, CF é útil para analisar capacidade de teste, manutenção, reutilização e propensão a erros e, por fim, MHF e AHF são úteis para avaliar manutenção.

No estudo relatado em (SASTRY; RAMESH; PADMAJA, 2011) é feita uma análise das métricas MOOD e Lorenz & Kidd, as quais são usadas para avaliar características dos componentes de software. Em relação às métricas MOOD, os autores observaram que as métricas AHF e MHF são úteis em avaliar funcionalidade e segurança, AIF, MIF, PF podem ser usadas para analisar reutilização, manutenção e capacidade de teste e, por fim, CF é capaz de avaliar reuso.

No trabalho apresentado em (ARORA et al., 2011), são avaliados os conjuntos de métricas MOOD, CK e QMOOD (BANSIYA; DAVIS, 2002). Foi realizado um estudo de caso para mostrar como essas métricas são úteis na análise da qualidade de software projetado usando paradigma orientado a objetos. Considerando as métricas do conjunto MOOD, os autores concluíram que MHF e AHF são capazes de avaliar a funcionalidade, MIF e AIF são úteis em analisar testabilidade, reutilização e funcionalidade do sistema e PF é adequada para verificar a reutilização.

Na pesquisa descrita em (SRINIVASAN; DEVI, 2014), é feita uma análise das métricas orientadas a objetos mais usadas na medição de software. Considerando as métricas MOOD, os autores concluíram que as métricas MIF e AIF são úteis em avaliar eficácia e manutenção do software e CF é capaz de auxiliar nas tarefas de manutenção e reusabilidade.

Na Tabela 5.2 são sumarizados os parâmetros de avaliação que podem ser utilizados na análise da qualidade de software orientado a objeto. Os parâmetros são baseados nos trabalhos

supracitados.

Tabela 5.2 – Uso de Métricas MOOD em Engenharia de Software

Parâmetros	**Métricas MOOD**					
	MIF	**AIF**	**MHF**	**AHF**	**PF**	**CF**
Reutilização	X	X	X	X	X	X
Capacidade de teste	X	X			X	X
Manutenção	X	X	X	X	X	X
Eficiência	X	X				
Funcionalidade	X	X	X	X		
Segurança			X	X		
Propensão a erros						X

5.2 Análise dos resultados

Nesta seção, os resultados serão analisados se estão dentro dos parâmetros descritos no Capítulo 3. Inicialmente, para cada conjunto de métricas, serão comparados os valores encontrados nos softwares utilizados nesta pesquisa com os valores expostos nas Seções 3.2 e 3.3. Posteriormente, será discutido de que maneira as métricas podem contribuir para o gerenciamento de projetos de software e a avaliação de equipes de desenvolvimento.

5.2.1 Avaliação dos softwares por meio das Métricas CK

Nesta seção, os resultados das métricas CK serão analisados baseando-se nos estudos apresentados em (JULIANO; TRAVENÇOLO; SOARES, 2014) e serão comparados com os valores da Tabela 3.1. Os softwares com as siglas de S1 a S4 foram desenvolvidos nas empresas, e os de ST1 a ST9 correspondem aos softwares produzidos pelos estudantes.

DIT: Árvores mais profundas constituem maior complexidade do projeto, visto que mais métodos e classes estão envolvidos. Quanto mais profunda for uma classe na hierarquia, maior a reutilização de métodos herdados. A classe em maior profundidade é mais especializada comparada à classe em menor profundidade na hierarquia de herança. A classe mais especializada oferece menos oportunidades de herdá-la (CHIDAMBER; KEMERER, 1994), (GILL; SIKKA, 2011).

Neste contexto, os softwares S1, S2 e S4 se encontram dentro dos padrões. No software S3 foi detectada uma classe com resultado 4 para DIT, sendo que a partir de 3 é considerado anomalia. Dos softwares dos estudantes, ST6, ST7 e ST9 estão no limite do valor considerado como anomalia, e ST1 alcançou o dobro desse valor.

NOC: Essa métrica avalia principalmente a eficiência, reutilização e capacidade de teste (GANDHI; BHATIA, 2010). Quanto maior o valor do NOC, menor o número de falhas.

No trabalho descrito em (ZHOU; LEUNG, 2006), foi observado que a métrica NOC é inversamente proporcional à propensão a erros. Quanto mais filhos uma classe possuir, maior é a importância dela no sistema, e possivelmente, mais testada ela foi. Neste experimento, no software S3 foi detectado valor acima do limite considerado como anomalia. Nos softwares desenvolvidos pelos estudantes, nenhuma anormalidade foi encontrada.

CBO: Acoplamento excessivo entre classes de objetos impede a reutilização. Quanto maior for o grau de acoplamento e a sensibilidade para alterações em outras partes do projeto, consequentemente a manutenção será mais difícil. Uma medida de acoplamento é útil para determinar a complexidade do teste de várias partes de um projeto, quanto maior o acoplamento de uma classe, mais rigoroso o teste precisa ser (CHIDAMBER; KEMERER, 1994).

Nesse contexto, no software S3 foi encontrado um valor muito acima ao da anomalia. Desta maneira, é possível afirmar que a manutenção desse software é uma tarefa difícil de realizar.

RFC: Se um grande número de métodos pode ser invocado em resposta a uma mensagem, o teste de depuração e da classe torna-se mais complicado, uma vez que requer um maior nível de compreensão necessária por parte do dispositivo de teste. Quanto maior o número de métodos que podem ser invocados a partir de uma classe, maior a sua complexidade (CHIDAMBER; KEMERER, 1994).

Todos os softwares produzidos nas empresas obtiveram valores superiores à anomalia (46). Os valores mais distantes foram encontrados nos softwares S2 e S3. O valor 48, sendo o mais próximo do valor considerado como anomalia foi obtido pelo software S1. Entre os estudantes, apenas no software ST1 foi encontrado valor maior que a da anomalia.

LCOM: A coesão de métodos de uma classe é desejável, uma vez que promove o encapsulamento. A falta de coesão implica que a classe deveria ser dividida em duas ou mais subclasses. Com a baixa coesão, a complexidade aumenta, crescendo a probabilidade de erros durante o processo de desenvolvimento (CHIDAMBER; KEMERER, 1994).

Todos os softwares analisados ficaram abaixo do valor considerado como anomalia. Porém, na pesquisa descrita em (JULIANO; TRAVENÇOLO; SOARES, 2014) ficou constatado que essa métrica é ineficiente para realizar algumas atividades como propensão a erros e análise de refatoração, mas pode ser usada para avaliar a funcionalidade do software.

WMC: O número e a complexidade de métodos envolvidos possibilitam ter uma previsão de quanto tempo e esforço será necessário para desenvolver e manter a classe. Quanto maior o número de métodos em uma classe, maior será o impacto potencial sobre as classes filhas, uma vez que estas vão herdar todos os métodos definidos na classe. Classes com um grande número de métodos são susceptíveis de ser uma aplicação específica, limitando a possibilidade de reutilização (CHIDAMBER; KEMERER, 1994), (GANDHI; BHATIA, 2010).

Todos os softwares desenvolvidos nas empresas obtiveram valores anômalos para essa

métrica. Os valores mais distantes foram encontrados nos softwares S2 e S3: 176 e 183, respectivamente. Entre os estudantes, foram detectadas anomalias nos softwares ST1, ST4, ST5 e ST6, sendo que em ST1 foi detectado o valor 190 para esta métrica.

Na Tabela 5.3 são mostrados os valores anômalos para as métricas CK encontrados nos softwares analisados nesta pesquisa.

Tabela 5.3 – Valores anômalos encontrados para as métricas CK

Métricas CK			
Métrica	**Anomalia**	**Anomalia identificada**	**Software**
DIT	≥ 3	4	S3
		6	ST1
NOC	≥ 8	172	S3
CBO	≥ 20	296	S3
RFC	≥ 84	137	S2
		132	S3
		148	ST1
LCOM	≥ 388	⊗	⊗
WMC	≥ 29	46	S1
		176	S2
		183	S3
		95	S4
		190	ST1
		95	ST4
		38	ST5
		40	ST6

5.2.2 Avaliação dos softwares por meio das Métricas MOOD

Nesta seção, os resultados das métricas MOOD serão analisados e comparados com os valores da Tabela 3.5 apresentada no Capítulo 3, Seção 3.3.

MIF: Valores altos indicam herança excessiva, conduzindo assim a um maior acoplamento e reduzindo a possibilidade de reutilização e valores baixos indicam falta de herança (GUPTA; BATRA et al., 2012).

Neste contexto, S3, ST8 e ST9 atingiram o valor da anomalia, nos softwares S1, S2, S4, ST1, ST2, ST3, ST4, ST5 foi identificado o valor zero para essa essa métrica, indicando pouco uso da herança.

AIF: Da mesma maneira que MIF, altos valores de AIF indicam o uso excedente de herança (GUPTA; BATRA et al., 2012). Para esta métrica, foram encontrados valores anômalos em S3, ST1 e ST8. Porém, nos softwares S1, S2, S4, ST2, ST4 e ST5 os resultados foram nulos.

MHF: Valores altos indicam que o software possui o maior número dos métodos privados,

o que compromete a funcionalidade e reutilização. Por outro lado, valores baixos indicam que prevalecem os métodos públicos, o que os torna desprotegidos (SASTRY; RAMESH; PADMAJA, 2011).

Dos softwares analisados, nenhum apresentou resultado anômalo, porém ST1 obteve um valor considerado alto, S3 e S4 apresentaram valores baixos e ST6, ST7, ST8, ST9 obtiveram resultados nulos.

AHF: Da mesma maneira que a métrica MHF, valores altos indicam que no software predominam os atributos privados, o que prejudica a funcionalidade e reutilização. Valores baixos indicam que a maior parte dos atributos é pública e estes estão vulneráveis (SASTRY; RAMESH; PADMAJA, 2011).

Dos softwares analisados, S1, S2, S3, S4, ST1, ST2, ST4, ST6 e ST7 atingiram o valor considerado como anomalia, ST8 obteve valor nulo, enquanto os demais resultaram em valores altos.

PF: Polimorfismo é uma palavra de origem grega que significa “muitas formas”. Quando aplicada ao paradigma orientado a objetos, significa a possibilidade de enviar uma mensagem sem saber qual será a forma da classe receptora. Todas as classes que podem receber a mensagem pertencem à mesma árvore de hierarquia de herança (ABREU; CARAPUÇA, 1994). O polimorfismo sugere que, em alguns casos, os métodos predominantes poderiam contribuir para reduzir a complexidade e, portanto, para tornar o sistema mais compreensível e mais fácil de manter (JASSIM; ALTAANI, 2013).

Nos softwares S1, S2, S4, ST1, ST2, ST3, ST4, ST5, ST6 e ST7 foram detectados valores anômalos para esta métrica. Com exceção de ST6 e ST7, todos os outros softwares alcançaram o valor máximo possível (1,00).

CF: À medida que o acoplamento entre classes aumenta, a densidade de defeitos e retrabalho também deve aumentar. Este resultado mostra que o acoplamento em sistemas de software tem um impacto negativo na qualidade do software e deve ser evitado durante o projeto (JASSIM; ALTAANI, 2013).

Os softwares S1, S4 e todos desenvolvidos pelos estudantes atingiram o valor considerado como anomalia para esta métrica.

Na Tabela 5.4 são apresentados os valores anômalos para as métricas MOOD encontrados nos softwares avaliados.

Tabela 5.4 – Valores anômalos encontrados para as métricas MOOD

Métricas MOOD			
Métrica	**Anomalia**	**Anomalia identificada**	**Software**
MIF	≥ 0,35	0,65	S3
		0,80	ST8
		0,48	ST9
AIF	≥ 0,37	0,37	S3
		0,71	ST1
		0,78	ST8
MHF	≥ 0,74	⊗	⊗
AHF	≥ 83	0,90	S1
		0,89	S2
		0,97	S3
		0,88	S4
		0,93	ST1
		0,90	ST2
		1,00	ST4
		1,00	ST6
		0,94	ST7
PF	≥ 0,14	1,00	S1
		1,00	S2
		1,00	S4
		1,00	ST1
		1,00	ST2
		1,00	ST3
		1,00	ST4
		1,00	ST5
		0,14	ST6
		0,24	ST7
CF	≥ 0,08	0,18	S1
		0,50	S2
		1,00	ST1
		0,87	ST2
		0,50	ST3
		1,00	ST4
		0,09	ST5
		0,29	ST6
		0,21	ST7
		0,80	ST8
		0,60	ST9

5.2.3 Métricas de Software vs Equipe de desenvolvimento

Nesta seção, serão discutidas as suposições apresentadas no Capítulo 4, Seção 4.1.2.

Suposição 1:

Os conjuntos de métricas CK e MOOD são semelhantes, mas possuem diferenças em determinados aspectos. As métricas CK se concentram nas classes, enquanto que MOOD analisa o projeto. Por exemplo, as métricas de acoplamento de CBO e CF estão bastante relacionadas. A métrica CBO fornece acoplamento no grau de classe e CF fornece acoplamento do sistema. Ambos os conjuntos não estão completos, mas quando utilizados juntos contribuem para o gerenciamento do projeto (SUBRAMANYAM; KRISHNAN, 2003), (MAO; JIANG, 2008).

Diante do exposto na Seção 5.1 e ao analisar as Tabelas 5.1 e 5.2, pode-se observar que as métricas aplicadas neste trabalho são capazes de avaliar características do software que são determinantes para a sua qualidade. Essas características incluem capacidade de teste, eficiência, reutilização, manutenção, falha no projeto de classes, tempo de desenvolvimento, propensão a erros, funcionalidade e segurança.

Aplicando as métricas durante o desenvolvimento do projeto, é possível avaliar em quais destas características o software deve ser melhorado e não descobrir apenas quando o projeto for finalizado. A aplicação periódica permite conhecer em quais critérios de qualidade o software precisa ser modificado, e ao manter o histórico dessas aplicações, torna-se possível conhecer sua evolução. Para avaliação periódica, as métricas podem ser aplicadas em módulos do sistema, e a cada integração, uma nova análise pode ser ser realizada. Os valores referência descritos nesse trabalho ajudarão os gerentes a analisar a qualidade do software em desenvolvimento, pois eles servem como parâmetros para serem usados na avaliação.

Desta forma, confirma-se a Suposição 1, evidenciando que métricas de produto de software são úteis no gerenciamento de projetos.

Suposição 2:

Considerando os resultados apresentados na Tabela 5.3, foi verificado que os piores resultados foram encontrados no software S3. Foram detectados valores anômalos para todas as métricas do conjunto CK, exceto LCOM. As informações mostradas nas tabelas da Seção 4.3.1 do Capítulo 4, mostram que os programadores P4 e P5 (desenvolvedores do software S3) possuem graduação, sem nenhuma certificação ou outra capacitação em andamento. Foi observado também que há uma diferença no tempo de experiência entre esses programadores. Enquanto o programador P5 tem uma experiência no intervalo entre 8 a 10 anos em tecnologia da informação e programação, o programador P4 possui entre 2 e 4 anos. Outra diferença está na experiência da linguagem Java. Enquanto P4 tem uma experiência de 4 anos, P5 trabalha com essa linguagem há 10 anos.

Em contrapartida, os softwares S1 e S4 obtiveram os melhores resultados. Ao observar a

escolaridade dos programadores, os responsáveis pelo software S1, P1 está cursando mestrado e P2 possui especialização. O mesmo acontece com P6, responsável pelo software S4, que está com o mestrado em andamento. Outro ponto importante diz respeito à experiência dos programadores. Os responsáveis pelo software S1 tem experiência mais próxima, tanto em relação ao tempo de trabalho em tecnologia da informação, programação e na linguagem Java.

Em relação às métricas MOOD, foram detectados valores anômalos em todos os softwares avaliados. No software ST1, apenas nas métricas MIF e MHF não foram identificados valores fora do padrão. Para a métrica CF, em todos os softwares desenvolvidos pelos estudantes foram encontrados valores anômalos. O melhor desempenho foi alcançado pelo estudante ST9, visto que foram constatados valores fora do padrão apenas em duas métricas (MIF e CF).

Com base nas informações descritas nas Subseções 5.2.1 e 5.2.2 foi feita uma análise geral dos softwares estudados. Na Tabela 5.5 são apresentados quais softwares não obtiveram bons resultados após a aplicação das métricas e em quais critérios necessitam melhorias. Os softwares inclusos na tabela obtiveram valores anômalos em pelo menos uma das métricas que avalia os critérios descritos na primeira coluna.

Tabela 5.5 – Avaliação geral dos softwares

Critério de avaliação	**Métricas relacionadas**		**Softwares com anomalias**	
	CK	**MOOD**	**Empresas**	**Alunos**
Capacidade de teste	DIT, NOC, CBO e RFC	MIF, AIF, PF e CF	Todos	Todos
Eficiência	DIT, NOC	MIF e AIF	S3	ST1, ST8, ST9
Reutilização	DIT, NOC e CBO	Todas	Todos	Todos
Manutenção	DIT, NOC, CBO, RFC e WMC	Todas	Todos	Todos
Tempo de desenvolvimento	WMC	Não há	Todos	ST1, ST4, ST5, ST6
Propensão a erros	Todas	CF	Todos	Todos
Funcionalidade	LCOM	MIF, AIF, MHF e AHF	Todos	ST1, ST2, ST4,ST6 ST7,ST8 e ST9
Segurança	Não há	MHF e AHF	Todos	ST1, ST2, ST4, ST6 e ST7

De posse da avaliação do software que está sendo desenvolvido, o gerente do projeto pode tomar decisões baseando-se nos critérios que afetam a qualidade do produto. Ao observar a Tabela 5.5, pode-se afirmar que todos os softwares desenvolvidos nas empresas apresentam problemas relativos à reutilização dos componentes, manutenção, tempo de desenvolvimento e propensão a erros, sendo estes dois últimos fatores críticos que afetam mais diretamente o relacionamento com o cliente. Ao realizar periodicamente essas avaliações, torna-se possível ao gerente conhecer sua equipe e perceber qual a melhor formação para determinados projetos.

Outra utilidade do uso de métricas de software na avaliação da equipe pode ser na seleção de novos integrantes. Supondo que os alunos participantes dessa pesquisa estivessem disputando uma vaga de emprego, ao observar os resultados das métricas ficaria constatado que os estudantes ST3 e ST9 possuem as melhores práticas de programação.

Diante dos resultados, há evidências que equipes de desenvolvimento de software podem ser avaliadas tecnicamente por meio de métricas de produto de software, confirmando-se a Suposição 2.

No Apêndice B é mostrado um exemplo de como a equipe de desenvolvimento e seus membros podem ser avaliados. Inicialmente, são registrados os dados individuais de cada desenvolvedor, onde procura-se saber sobre sua escolaridade, experiência e projetos anteriores. Em seguida, é realizada a aplicação das métricas, que pode ser feita no software completo ou apenas em determinado módulo em que o desenvolvedor atuou. Para melhor entendimento, podem ser incluídas também informações sobre o projeto, tais como prazo, reuniões e alterações na equipe. Esses dados são capazes de ajudar a entender possíveis falhas do projeto, visto que as alterações na equipe podem interferir na qualidade do produto final. Essa avaliação pode ser feita periodicamente com o objetivo de ter um controle sobre o projeto e permitir ao gerente que conheça sua equipe, o que pode ser útil em projetos futuros.

6 Conclusão

Construir um software com qualidade é uma preocupação presente em todo o processo de desenvolvimento. O software deve satisfazer às necessidades dos clientes, incluindo a conformidade com os requisitos e o cumprimento de custos e prazos. Assim, é preciso um gerenciamento eficaz desde a concepção do projeto até a manutenção do software.

A equipe de desenvolvimento é composta por pessoas com diferentes formações, experiências e hábitos de programação. O desempenho da equipe está relacionado tanto com a interação entre os membros, quanto com as práticas adotadas pela empresa. A avaliação do esforço individual e da equipe pelo gerente é necessária para garantir a qualidade do projeto.

Avaliar a equipe não deve ser uma tarefa subjetiva, é preciso usar métodos concretos que reflitam o real contexto do grupo. Com esse pressuposto, o objetivo deste trabalho foi avaliar equipes e indivíduos analisando o desempenho atual por meio do software produzido.

Com o propósito de alcançar esse objetivo, nesse trabalho foram aplicados como instrumentos de pesquisa a revisão da literatura e o estudo de caso.

A revisão da literatura propiciou dois resultados. O primeiro foi conhecer métricas específicas de projetos orientados a objeto e como essas métricas foram utlizadas em seu gerenciamento. O segundo foi descobrir valores referência para as métricas aplicadas, com a finalidade de obter critérios que serviram de base para fazer a comparação em relação a outros softwares desenvolvidos. Estes resultados foram apresentados na Seção 1.4 do Capítulo 1 e no Capítulo 3, respectivamente.

Após obter esses conhecimentos, foi aplicado o estudo de caso em duas empresas públicas locais e em alunos de graduação de duas universidades locais. Ao todo, treze softwares foram avaliados, sendo quatro desenvolvidos pelas empresas e nove pelos estudantes. A descrição do estudo de caso foi mostrada no Capítulo 4.

Neste trabalho, as métricas de software foram usadas para avaliar equipes e indivíduos analisando o desempenho atual e perspectivas de desempenho futuro do ponto de vista gerencial em um projeto de desenvolvimento de software. Para cumprir esse objetivo, foram coletadas informações sobre os participantes e calculadas as métricas dos softwares produzidos por eles. As métricas dos conjuntos CK e MOOD, apresentados no Capítulo 2, foram aplicadas aos softwares estudados.

Os resultados foram analisados se estão dentro dos parâmetros descritos nas Seções 3.2 e 3.3 do Capítulo 3. No Capítulo 5 é apresentada a análise dos resultados obtidos por meio da execução do estudo de caso.

6.1 Respostas às questões de pesquisa

As questões de pesquisa propostas nesse trabalho foram respondidas a partir dos resultados obtidos na execução do estudo de caso.

Q1: Métricas de software orientadas a objetos podem ser usadas para avaliar a qualidade do produto final do ponto de vista do gerenciamento de projetos?

A aplicação de métricas mostrou-se útil em avaliar o software durante o desenvolvimento do projeto. Foi observado que características que analisam a qualidade do software podem ser avaliadas por meio de métricas. Como exemplo dessas características, podemos citar capacidade de teste, reutilização, manutenção, tempo de desenvolvimento, eficiência, propensão a erros, funcionalidade e segurança.

As métricas utilizadas na pesquisa, quando aplicadas em conjunto, contribuem para a análise destas características. Elas são úteis na previsão de quais destes pontos o produto de software deve ser melhorado e não descobrir apenas quando o produto for finalizado. Como exemplo, foi observado que as métricas DIT, NOC, CBO e RFC do conjunto CK e as métricas MIF, AIF, PF e CF do conjunto MOOD, ao serem aplicadas concomitantemente, avaliam a capacidade de teste do software.

Desta forma, diante dos resultados encontrados, evidencia-se que métricas de software orientadas a objetos são úteis no gerenciamento de projetos.

Q2: É possível avaliar tecnicamente equipes de desenvolvimento de software por meio de métricas de produto de software?

Por meio da avaliação do software em desenvolvimento, o gerente do projeto pode tomar decisões ao analisar os critérios que afetam a qualidade do produto. Ao realizar periodicamente essas avaliações, é possível que o gerente coordene sua equipe e compreenda qual a melhor formação para projetos específicos.

Nesse sentido, as métricas de software podem auxiliar nas decisões que causam mudanças na equipe, tais como escolher o novo gerente do projeto, aumentar salários na equipe e também como investir em capacitação e até mesmo na seleção de novos contratados. Por exemplo, ao avaliar dois ou mais candidatos, e tendo em vista os critérios de qualidade mais importantes, pode-se usar uma determinada métrica para avaliar um produto de software feito pelos candidatos.

Diante do exposto, confirma-se evidências de que equipes de desenvolvimento de software podem ser avaliadas tecnicamente por meio de métricas de produto de software.

6.2 Contribuições

As métricas do paradigma orientado a objetos são aplicadas para acompanhar o processo de desenvolvimento de software, porém com o intuito de avaliar a qualidade do produto final. Neste trabalho, o uso de métricas de software contribuiu para evidenciar que as métricas também podem ser usadas na tomada de decisão no gerenciamento de projetos, ao considerar a qualificação técnica dos desenvolvedores.

A principal contribuição do trabalho foi mostrar como a equipe de desenvolvimento pode ser avaliada tecnicamente por meio do software produzido. Esse tipo de avaliação é importante para a gerência de projetos, pois auxilia na detecção e administração de possíveis falhas. Quanto mais cedo as métricas forem aplicadas e interpretadas, mais úteis serão na melhoria do processo de desenvolvimento.

Outra contribuição importante foi a definição dos valores referência para as métricas MOOD, apresentados na Seção 3.3 do Capítulo 3. Esses valores são importantes para que os softwares sejam avaliados seguindo parâmetros definidos. Podem ser utilizados para identificação de prováveis problemas no software e, assim, melhorar a sua qualidade. Esses valores são úteis para interpretar a complexidade do projeto, pois os riscos do projeto podem ser avaliados por meio deles.

Este trabalho também resultou em quatro trabalhos em congressos científicos:

1. (BARROSO et al., 2016). ***An Evaluation of Influence of Human Personality in Software Development: An Experience Report.*** publicado na 8th Euro American Conference on Telematics and Information Systems (EATIS 2016), conferência com Qualis B4, em Abril de 2016;

2. Madureira, J. S., Barroso, A. S., Soares, M. S., e do Nascimento, R. P. ***An Experiment to Evaluate Software Development Teams by using Object-Oriented Metrics*** submetido na 17th International Conference on Computational Science and Its Applications (ICCSA 2017) conferência com Qualis B1, em Fevereiro de 2017;

3. Barroso, A. S., Madureira, J. S., Soares, M. S., e do Nascimento, R. P. ***Relationship between Personality Traits and Software Quality - Big Five Model vs. Object-oriented Software Metrics*** aceito na 19th International Conference on Enterprise Information Systems (ICEIS 2017), conferência com Qualis B1.

4. Barroso, A. S., Madureira, J. S., Soares, M. S., e do Nascimento, R. P. ***Influence of Human Personality in Software Engineering - A Systematic Literature Review*** aceito na 19th International Conference on Enterprise Information Systems (ICEIS 2017), conferência com Qualis B1.

6.3 Trabalhos futuros

Para continuidade e aprimoramento desta pesquisa, trabalhos futuros podem ser realizados. Como exemplos, podem ser citados:

1. **Novo estudo de caso:** para o novo estudo, sugere-se a coleta de informações desde o início do projeto que possam contribuir para melhor entendimento do sistema, tais como:
    a) Prazo concedido para execução do projeto;
    b) Periodicidade das reuniões acerca do projeto;
    c) Possíveis mudanças na equipe de desenvolvimento;
    d) Investimento da empresa com capacitação;
    e) Aplicação periódica das métricas.
2. **Ferramenta de apoio no gerenciamento de projetos**: sugere-se o desenvolvimento de uma ferramenta tecnológica que execute a coleta das métricas e apresente ao desenvolvedor as possíveis falhas do projeto, como por exemplo, complexidades relativas a:
    a) Manutenção;
    b) Teste;
    c) Reutilização;
    d) Segurança;
    e) Eficiência, dentre outras características.

Referências

ABREU, F. B.; CARAPUÇA, R. Object-Oriented Software Engineering: Measuring and Controlling the Development Process. *Proceedings of the 4th International Conference on Software Quality*, v. 186, p. 1–8, 1994.

ABUASAD, A.; ALSMADI, I. M. Evaluating the Correlation between Software Defect and Design Coupling Metrics. *2012 International Conference on Computer, Information and Telecommunication Systems (CITS)*, p. 1–5, 2012.

ACUÑA, S. T.; JURISTO, N.; MORENO, A. M. Emphasizing Human Capabilities in Software Development. *IEEE Computer Society*, v. 23, n. 2, p. 94–101, 2006.

AHMED, F.; CAPRETZ, L. F.; CAMPBELL, P. Evaluating the Demand for Soft Skills in Software Development. *IT Professional*, IEEE Computer Society, v. 14, n. 1, p. 44–49, 2012.

ALBRECHT, A. J.; GAFFNEY, J. E. Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, IEEE, n. 6, p. 639–648, 1983.

ALVES, T. L.; YPMA, C.; VISSER, J. Deriving Metric Thresholds from Benchmark Data. *26th IEEE International Conference on Software Maintenance (ICSM)*, p. 1–10, 2010.

ANDRES, H. P. A Comparison of Face-to-Face and Virtual Software Development Teams. *Team Performance Management: An International Journal*, v. 8, n. 1/2, p. 39–48, 2002.

ARORA, D. et al. Software Quality Estimation Through Object Oriented Design Metrics. *Int. J. Computer Science and Network Security*, v. 11, n. 4, p. 100–104, 2011.

BANSIYA, J.; DAVIS, C. G. A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Transactions on Software Engineering*, IEEE, v. 28, n. 1, p. 4–17, 2002.

BARROSO, A. S. et al. An Evaluation of Influence of Human Personality in Software Development: An Experience Report. *8th Euro American Conference on Telematics and Information Systems (EATIS), 2016*, p. 1–6, 2016.

BASILI, V.; WEISS, D. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, Institute of Electrical and Electronics Engineers, v. 10, n. 6, p. 728–738, 1984.

BEHKAMAL, B.; KAHANI, M.; AKBARI, M. K. Customizing ISO 9126 Quality Model for Evaluation of B2B applications. *Information and Software Technology*, Elsevier, v. 51, n. 3, p. 599–609, 2009.

BENESTAD, H. C.; ANDA, B.; ARISHOLM, E. Assessing Software Product Maintainability Based on Class-Level Structural Measures. *International Conference on Product Focused Software Process Improvement*, p. 94–111, 2006.

BENLARBI, S. et al. Thresholds for Object-Oriented Measures. *11th International Symposium on Software Reliability Engineering, 2000. ISSRE 2000. Proceedings*, p. 24–38, 2000.

BRAINS J. IntelliJ Idea. 2016. Disponível em: <https://www.jetbrains.com/idea/>.

CARPENTER, M. A. The Implications of Strategy and Social Context for the Relationship between Top Management Team Heterogeneity and Firm Performance. *Strategic Management Journal*, Wiley Online Library, v. 23, n. 3, p. 275–284, 2002.

CHIDAMBER, S. R.; KEMERER, C. F. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, v. 20, n. 6, p. 476–493, 1994.

COLOMO-PALACIOS, R. et al. Analyzing Human Resource Management Practices within the GSD Context. *Journal of Global Information Technology Management*, v. 15, n. 3, p. 30–54, 2012.

DINGSØYR, T.; DYBÅ, T. Team Effectiveness in Software Development: Human and Cooperative Aspects in Team Effectiveness Models and Priorities for Future Studies. *Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering*, p. 27–29, 2012.

DUBEY, S. K.; RANA, A. A Comprehensive Assessment of Object-Oriented Software Systems Using Metrics Approach. *International Journal on Computer Science and Engineering*, Citeseer, v. 2, n. 08, p. 2726–2730, 2010.

DUBEY, S. K.; SHARMA, A. et al. Comparison Study and Review on Object-Oriented Metrics. *Global Journal of Computer Science and Technology*, v. 12, n. 7, p. 39–48, 2012.

EL-LATEEF, T. A.; YOUSEF, A.; ISMAIL, M. Object Oriented Design Metrics Framework Based on Code Extraction. *International Conference on Computer Engineering & Systems*, ICCES 2008, p. 291–295, 2008.

ELISH, M. O.; AL-YAFEI, A. H.; AL-MULHEM, M. Empirical Comparison of Three Metrics suites for Fault Prediction in Packages of Object-Oriented Systems: A Case Study of Eclipse. *Advances in Engineering Software*, Elsevier, v. 42, n. 10, p. 852–859, 2011.

FARAJ, S.; SPROULL, L. Coordinating expertise in software development teams. *Management science*, INFORMS, v. 46, n. 12, p. 1554–1568, 2000.

FERREIRA, K. A. et al. Identifying Thresholds for Object-Oriented Software Metrics. *Journal of Systems and Software*, Elsevier, v. 85, n. 2, p. 244–257, 2012.

FISHER, E. What Practitioners Consider to be the Skills and Behaviours of an Effective People Project manager. *International Journal of Project Management*, Elsevier, v. 29, n. 8, p. 994–1002, 2011.

GALINEC, D. Human Capital Management Process Based on Information Technology Models and Governance. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, IGI Global, v. 1, n. 1, p. 44–60, 2010.

GANDHI, P.; BHATIA, P. K. Reusability Metrics for Object-Oriented System: An Alternative Approach. *International Journal of Software Engineering (IJSE)*, v. 1, n. 4, p. 63–72, 2010.

GENERO, M.; PIATTINI, M.; CALERO, C. A Survey of Metrics for UML Class Diagrams. *Journal of Object Technology*, v. 4, n. 9, p. 59–92, 2005.

GILL, N. S.; SIKKA, S. Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD. *International Journal on Computer Science and Engineering*, Engg Journals Publications, v. 3, n. 6, p. 2300–2309, 2011.

GÓMEZ, M.; ACUÑA, S. T. Study of the Relationships between Personality, Satisfaction and Product Quality in Software Development Teams. *SEKE*, Citeseer, p. 292–295, 2007.

GUPTA, A.; BATRA, G. et al. Analyzing Theoretical Basis and Inconsistencies of Object Oriented Metrics. *International Journal on Computer Science and Engineering*, Engg Journals Publications, v. 4, n. 5, p. 803–808, 2012.

GUZZO, R. A.; DICKSON, M. W. Teams in Organizations: Recent Research on Performance and Effectiveness. *Annual Review of Psychology*, v. 47, n. 1, p. 307–338, 1996.

HALL, K. Jhawk 5. 2016. Disponível em: <http://www.virtualmachinery.com/jhawkprod.htm>.

HALSTEAD, M. H. Elements of Software Science. Elsevier New York, v. 7, 1977.

HARRISON, R.; COUNSELL, S. J.; NITHI, R. V. An Evaluation of the MOOD Set of Object-Oriented Software Metrics. *IEEE Transactions on Software Engineering*, IEEE, v. 24, n. 6, p. 491–496, 1998.

HODA, R.; NOBLE, J.; MARSHALL, S. Self-Organizing Roles on Agile Software Development Teams. *IEEE Transactions on Software Engineering*, IEEE, v. 39, n. 3, p. 422–444, 2013.

HOEGL, M.; GEMUENDEN, H. G. Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence. *Organization science*, INFORMS, v. 12, n. 4, p. 435–449, 2001.

HUCK, S. W. Reading Statistics and Research. Pearson, 2012.

HUDA, M.; ARYA, Y.; KHAN, M. Evaluating Effectiveness Factor of Object Oriented Design: A Testability Perspective. *International Journal of Software Engineering & Applications*, Academy & Industry Research Collaboration Center (AIRCC), v. 6, n. 1, p. 41–49, 2015.

HÜLSHEGER, U. R.; ANDERSON, N.; SALGADO, J. F. Team-Level Predictors of Innovation at Work: A Comprehensive Meta-Analysis Spanning Three Decades of Research. *Journal of Applied psychology*, American Psychological Association, v. 94, n. 5, p. 1128–1145, 2009.

HYATT, D. E.; RUDDY, T. M. An Examination of the Relationship between Work Group Characteristics and Performance: Once More into the Breech. *Personnel Psychology*, Wiley Online Library, v. 50, n. 3, p. 553–585, 1997.

IQBAL, S.; NAEEM, M.; KHAN, A. Yet Another Set of Requirement Metrics for Software Projects. *International Journal of Software Engineering and Its Applications*, Citeseer, v. 6, n. 1, p. 19–28, 2012.

ISO. IEC 25000 Software and System Engineering–Software Product Quality Requirements and Evaluation (SQuaRE)–Guide to SQuaRE. *International Organization for Standarization*, 2005.

JASSIM, F.; ALTAANI, F. Statistical Approach for Predicting Factors of Mood Method for Object Oriented. *International Journal of Computer Science Issues*, v. 10, n. 1, p. 589–593, 2013.

JOHARI, K.; KAUR, A. Validation of Object Oriented Metrics Using Open Source Software System: An Empirical Study. *ACM SIGSOFT Software Engineering Notes*, ACM, v. 37, n. 1, p. 1–4, 2012.

JORDAN, M. H.; FEILD, H. S.; ARMENAKIS, A. A. The Relationship of Group Process Variables and Team Performance a Team-Level Analysis in a Field Setting. *Small Group Research*, Sage Publications, v. 33, n. 1, p. 121–150, 2002.

JULIANO, R. C.; TRAVENÇOLO, B. A. N.; SOARES, M. S. Detection of Software Anomalies Using Object-oriented Metrics. *Proceedings of the 16th International Conference on Enterprise Information Systems - ICEIS*, v. 2, p. 241–248, 2014.

JUNG, H.-W. Validating the External Quality Subcharacteristics of Software Products According to ISO/IEC 9126. *Computer Standards & Interfaces*, Elsevier, v. 29, n. 6, p. 653–661, 2007.

KAKARONTZAS, G. et al. Layer Assessment of Object-Oriented Software: A Mmetric Facilitating White-Box Reuse. *Journal of Systems and Software*, Elsevier, v. 86, n. 2, p. 349–366, 2013.

KAUR, A. et al. Empirical Analysis of CK & MOOD Metric Suit. *Int. Journal of Innovation, Management and Technology*, v. 1, n. 5, p. 447–452, 2010.

KAUR, S.; SINGH, S.; KAUR, H. A Quantitative Investigation of Software Metrics Threshold Values at Acceptable Risk Level. *International Journal of Engineering Research and Technology*, v. 2, n. 3, p. 1–7, 2013.

KOSTOPOULOS, K. C.; BOZIONELOS, N. Team Exploratory and Exploitative Learning: Psychological Safety, Task Conflict and Team Performance. *Group & Organization Management*, v. 36, n. 3, p. 385–415, 2011.

KRISHNAIAH, R.; PRASAD, B. S. Analysis of Object Oriented Metrics. *International Journal of Computational Engineering Research*, v. 2, n. 5, p. 1474–1479, 2012.

LALSING, V.; KISHNAH, S.; PUDARUTH, S. People Factors in Agile Software Development and Project Management. *International Journal of Software Engineering & Applications*, Academy & Industry Research Collaboration Center (AIRCC), v. 3, n. 1, p. 117–137, 2012.

LEPINE, J. A. Team Adaptation and Postchange Performance: Effects of Team Composition in Terms of Members' Cognitive Ability and Personality. *Journal of Applied Psychology*, American Psychological Association, v. 88, n. 1, p. 27–39, 2003.

LEPINE, J. A. et al. A Meta-Analysis of Teamwork Processes: Tests of a Multidimensional Model and Relationships with Team Effectiveness Criteria. *Personnel Psychology*, Wiley Online Library, v. 61, n. 2, p. 273–307, 2008.

LESSMANN, S. et al. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, IEEE, v. 34, n. 4, p. 485–496, 2008.

LI, W.; HENRY, S. Object-Oriented Metrics that Predict Maintainability. *Journal of systems and software*, Elsevier, v. 23, n. 2, p. 111–122, 1993.

LIMA, E. de C.; RESENDE, A. M. P. de; LETHBRIDGE, T. C. The Uncomfortable Discrepancies of Software Metric Thresholds and Reference Values in Literature. *ICSEA 2016*, p. 1–9, 2016.

LITTLEFAIR, T.; WATKINS, K.; MARCUS, R. CCCC C and C++ Code Counter. 2013. Disponível em: <https://sourceforge.net/projects/cccc/>.

LORENZ, M.; KIDD, J. Object-Oriented Software Metrics: A Practical Guide. Prentice-Hall, Inc., 1994.

LOUGHRY, M. L.; OHLAND, M. W.; MOORE, D. D. Development of a Theory-Based Assessment of Team Member Effectiveness. *Educational and Psychological Measurement*, Sage Publications, v. 67, n. 3, p. 505–524, 2007.

MA, Y. et al. A Complexity Metrics Set for Large-Scale Object-Oriented Software Systems. *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, p. 189–195, 2006.

MAO, M.; JIANG, Y. A Coherent Object-Oriented (OO) Software Metric Framework Model: Software Engineering. *Computer Science and Software Engineering, 2008 International Conference on*, v. 2, p. 68–72, 2008.

MARINESCU, R.; LANZA, M. Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer-Verlag, 2006.

MARTIN, R. OO Design Quality Metrics. *An Analysis of Dependencies*, v. 12, p. 151–170, 1994.

MATUSSE, E. A.; HUZITA, E. H. M.; TAIT, T. F. C. Metrics and Indicators to Assist in the Distribution of Process Steps for Distributed Software Development: A Systematic Review. *XXXVIII Conferencia Latinoamericana en Informatica (CLEI)*, p. 1–10, 2012.

MCCABE, T. J. A Complexity Measure. *IEEE Transactions on Software Engineering*, IEEE, n. 4, p. 308–320, 1976.

MENZIES, T.; GREENWALD, J.; FRANK, A. Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, IEEE, v. 33, n. 1, p. 2–13, 2007.

MOE, N. B.; DINGSØYR, T.; DYBÅ, T. A Teamwork Model for Understanding an Agile Team: A Case Study of a Scrum Project. *Information and Software Technology*, Elsevier, v. 52, n. 5, p. 480–491, 2010.

MOSER, R.; PEDRYCZ, W.; SUCCI, G. A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction. *Proceedings of the 30th International Conference on Software Engineering*, p. 181–190, 2008.

NAIR, T. G.; SELVARANI, R. Defect Proneness Estimation and Feedback Approach for Software Design Quality Improvement. *Information and software technology*, Elsevier, v. 54, n. 3, p. 274–285, 2012.

NAIR, T. G.; SUMA, V.; TIWARI, P. K. Significance of Depth of Inspection and Inspection Performance Metrics for Consistent Defect Management in Software Industry. *IET Software*, v. 6, n. 6, p. 524–535, 2012.

NIENABER, R. C.; BARNARD, A. A Generic Agent Framework to Support the Various Software Project Management Processes. *Interdisciplinary Journal of Information, Knowledge, and Management*, Informing Science Institute, v. 2, p. 149–162, 2007.

OLAGUE, H. M. et al. Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. *IEEE Transactions on Software Engineering*, IEEE, v. 33, n. 6, p. 402–419, 2007.

OLBRICH, S. et al. The Evolution and Impact of Code Smells: A Case Study of Two Open Source Systems. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, p. 390–400, 2009.

PERRY, D. E.; SIM, S. E.; EASTERBROOK, S. M. Case Studies for Software Engineers. *ICSE 2004. Proceedings. 26th International Conference on Software Engineering*, p. 736–738, 2004.

RADJENOVIĆ, D. et al. Software Fault Prediction Metrics: A Systematic Literature Review. *Information and Software Technology*, Elsevier, v. 55, n. 8, p. 1397–1418, 2013.

RIAZ, M.; MENDES, E.; TEMPERO, E. A Systematic Review of Software Maintainability Prediction and Metrics. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, p. 367–377, 2009.

ROBLES, G. et al. Modification and Developer Metrics at the Function Level: Metrics for the Study of the Evolution of a Software Project. *3rd International Workshop on Emerging Trends in Software Metrics (WETSoM),*, p. 49–55, 2012.

ROSEN, M. A. et al. Tools for Evaluating Team Performance in Simulation-Based Training. *Journal of Emergencies, Trauma, and Shock*, Medknow Publications, v. 3, n. 4, p. 353–359, 2010.

RUNESON, P.; HÖST, M. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, Springer, v. 14, n. 2, p. 131–164, 2009.

RYAN, S.; O'CONNOR, R. V. Development of a Team Measure for Tacit Knowledge in Software Development Teams. *Journal of Systems and Software*, Elsevier, v. 82, n. 2, p. 229–240, 2009.

SASTRY, J.; RAMESH, K.; PADMAJA, M. Measuring Object-Oriented Systems Based on the Experimental Analysis of the Complexity Metrics. *International Journal of Engineering Science and Technology*, v. 1, n. 3, p. 3726–3731, 2011.

SHARMA, A. K.; KALIA, A.; SINGH, H. Metrics Identification for Measuring Object Oriented Software Quality. *International Journal of Soft Computing and Engineering*, v. 2, n. 5, p. 255–258, 2012.

SHATNAWI, R. A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems. *IEEE Transactions on Software Engineering*, IEEE, v. 36, n. 2, p. 216–225, 2010.

SHATNAWI, R.; LI, W. The Effectiveness of Software Metrics in Identifying Error-Prone Classes in Post-Release Software Evolution Process. *Journal of systems and software*, Elsevier, v. 81, n. 11, p. 1868–1882, 2008.

SHATNAWI, R. et al. Finding Software Metrics Threshold Values Using ROC Curves. *Journal of Software Maintenance and Evolution: Research and Practice*, Wiley Online Library, v. 22, n. 1, p. 1–16, 2010.

SHELDON, F. T.; JERATH, K.; CHUNG, H. Metrics for Maintainability of Class Inheritance Hierarchies. *Journal of Software Maintenance and Evolution: Research and Practice*, v. 14, n. 3, p. 147–160, 2002.

SINGH, Y.; KAUR, A.; MALHOTRA, R. Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness Models. *Software Quality Journal*, Springer, v. 18, n. 1, p. 3–35, 2010.

SONG, Q. et al. A General Software Defect-Proneness Prediction Framework. *IEEE Transactions on Software Engineering*, IEEE, v. 37, n. 3, p. 356–370, 2011.

SPINELLIS, D. CKJM Chidamber and Kemerer Metrics Software. *v 1.6. Technical Report, Athens University of Economics and Business, Disponível em <http://-www. spinellis. gr/sw/ckjm>*, 2005.

SRINIVASAN, K.; DEVI, T. A Comprehensive Review and Analysis on Object-Oriented Software Metrics in Software Measurement. *International Journal on Computer Science and Engineering*, Engg Journals Publications, v. 6, n. 7, p. 247–261, 2014.

SRIVASTAVA, S.; KUMAR, R. Indirect Method to Measure Software Quality Using CK-OO Suite. *International Conference on Intelligent Systems and Signal Processing (ISSP)*, p. 47–51, 2013.

STEVENS, M. J.; CAMPION, M. A. Staffing Work Teams: Development and Validation of a Selection Test for Teamwork Settings. *Journal of Management*, Sage Publications, v. 25, n. 2, p. 207–228, 1999.

STROGGYLOS, K.; SPINELLIS, D. Refactoring–Does It Improve Software Quality? *Proceedings of the 5th International Workshop on Software Quality*, p. 10–17, 2007.

SUBRAMANYAM, R.; KRISHNAN, M. S. Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. *IEEE Transactions on Software Engineering*, IEEE, v. 29, n. 4, p. 297–310, 2003.

SUKOCO, A.; CUCUS, A. et al. Concept of Quality Measurement System Software Based on Standard ISO 9126 and ISO 19011. *2nd International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE)*, p. 105–108, 2012.

SURESH, Y.; PATI, J.; RATH, S. K. Effectiveness of Software Metrics for Object-Oriented System. *Procedia Technology*, Elsevier, v. 6, p. 420–427, 2012.

WALWORTH, T.; YEARWORTH, M.; SHRIEVES, L. Knowledge Management for Metrics: Enabling Analysis and Dissemination of Metrics. *8th Annual IEEE Systems Conference (SysCon)*, p. 199–205, 2014.

WHITMIRE, S. A. Object Oriented Design Measurement. *John Wiley & Sons, Inc.*, 1997.

WIESCHE, M.; KRCMAR, H. The Relationship of Personality Models and Development Tasks in Software Engineering. *Proceedings of the 52nd ACM conference on Computers and People Research*, p. 149–161, 2014.

WILKIE, F. G.; KITCHENHAM, B. A. Coupling Measures and Change Ripples in C++ Application Software. *Journal of Systems and Software*, Elsevier, v. 52, n. 2, p. 157–164, 2000.

WOHLIN, C. et al. Experimentation in Software Engineering. *Springer Science & Business Media*, 2012.

YILMAZ, M. et al. An Examination of Personality Traits and How They Impact on Software Development Teams. *Information and Software Technology*, Elsevier, p. 101–122, 2017.

ZHOU, Y.; LEUNG, H. Empirical Analysis of Object-Oriented Oesign Metrics for Predicting High and Low Severity Faults. *IEEE Transactions on Software Engineering*, IEEE, v. 32, n. 10, p. 771–789, 2006.

ZWEIG, M. H.; CAMPBELL, G. Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine. *Clinical Chemistry*, Clinical Chemistry, v. 39, n. 4, p. 561–577, 1993.

Apêndices

APÊNDICE A – Questionário aplicado na equipe de desenvolvimento

Informações sobre equipe de desenvolvimento de software

Questionário utilizado na coleta de informações sobre a equipe de desenvolvimento de software, para desenvolvimento do trabalho de mestrado no curso de Ciência da Computação da Universidade Federal de Sergipe (UFS).

*Obrigatório

Informe seu primeiro nome: *

Sua resposta

Qual a sua idade? *

Sua resposta

Qual o seu grau de escolaridade? *

- Nível médio
- Graduação
- Especialização
- Mestrado
- Doutorado

Possui alguma certificação? Qual (is)?

Sua resposta

Está cursando alguma capacitação na área de informática?

- Especialização
- Mestrado
- Doutorado
- Outro:

Há quanto tempo trabalha na área de Tecnologia da Informação? *

- Até 1 ano
- Entre 2 e 4 anos
- Entre 5 e 7 anos
- Entre 8 e 10 anos
- Mais de 10 anos

Há quanto tempo trabalha com programação? *

- Até 1 ano
- Entre 2 e 4 anos
- Entre 5 e 7 anos
- Entre 8 e 10 anos
- Mais de 10 anos

Há quanto tempo trabalha nesta empresa? *

- Até 1 ano
- Entre 2 e 4 anos
- Entre 5 e 7 anos
- Entre 8 e 10 anos
- Mais de 10 anos

Qual o seu tempo de experiência com a linguagem Java? *

Sua resposta

APÊNDICE B – Guia para avaliação de desenvolvedores

Nome da Empresa	
Avaliação Individual	
Data:	/ /
Projeto atual:	
Desenvolvedor (a):	

1. **Dados pessoais do desenvolvedor:**

   (a) **Nome completo:**

   (b) **Data de admissão:**

   (c) **Escolaridade** (Informar curso:)

   (d) **Capacitação em andamento:** (Se sim, informar curso:)

   (e) **Tempo de experiência no mercado de TI:**

   (f) **Tempo de experiência em programação:**

   (g) **Tempo de experiência com a linguagem do projeto atual:**

2. **Atividades na empresa:**

   (a) **Projetos anteriores:**

   (b) **Já gerenciou algum projeto anteriormente?**
   ( ) Não
   ( ) Sim (especificar:)

3. **Aplicação de métricas:**

( ) Software completo ( ) Módulo: _____

Data: / /

(a) **Métricas MOOD:**

Métrica	Valores				Valor encontrado
	Baixo	Normal	Alto	Anomalia	
MIF	$\leq$ 0,10	0,11 - 0,29	0,30 - 0,34	$\geq$ 0,35	
AIF	$\leq$ 0,12	0,13 - 0,31	0,32 - 0,36	$\geq$ 0,37	
MHF	$\leq$ 0,09	0,10 - 0,60	0,61 - 0,73	$\geq$ 0,74	
AHF	$\leq$ 0,10	0,11 - 0,67	0,68 - 0,82	$\geq$ 0,83	
PF	$\leq$ 0,02	0,03 - 0,11	0,12 - 0,13	$\geq$ 0,14	
CF	$\leq$ 0,03	0,04 - 0,06	0,07	$\geq$ 0,08	

(b) **Métricas CK:**

Métrica	Valores				Valor encontrado
	Baixo	Normal	Alto	Anomalia	
DIT	0	1	2	3	Mínimo: Máximo: Média:
NOC	0	1 - 5	6 - 7	8	Mínimo: Máximo: Média
CBO	2	3 - 14	15 - 19	20	Mínimo: Máximo: Média:
RFC	0	1 - 64	65 - 83	84	Mínimo: Máximo: Média
LCOM	0	1 - 297	298 - 387	388	Mínimo: Máximo: Média
WMC	1	2 - 21	22 - 28	29	Mínimo: Máximo: Média

4. **Considerações a acrescentar sobre o desenvolvedor:**

Dados do projeto:

1. **Data de início:** / /
2. **Data prevista para conclusão:** / /
3. **Periodicidade das reuniões:** Semanal ( ) Quinzenal ( ) Mensal ( ) Outra: ( ) _____
4. **Data da última reunião:** / /
5. **Participantes do projeto:**
   Nome: _____ Função: _____
   Nome: _____ Função: _____
   Nome: _____ Função: _____
   Nome: _____ Função: _____
   Nome: _____ Função: _____
6. **Alterações na equipe:**
   Nome: _____ Ingresso ( ) Saída ( ) Outra: ( ) _____
   Data: / /
   Nome: _____ Ingresso ( ) Saída ( ) Outra: ( ) _____
   Data: / /
   Nome: _____ Ingresso ( ) Saída ( ) Outra: ( ) _____
   Data: / /
7. **Dados da última coleta das métricas:**
   ( ) Software completo ( ) Módulo: _____
   Data: / /

Tabela 1: Métricas CK

Métrica	Valor encontrado
DIT	Mínimo: Máximo: Média:
NOC	Mínimo: Máximo: Média
CBO	Mínimo: Máximo: Média:
RFC	Mínimo: Máximo: Média
LCOM	Mínimo: Máximo: Média
WMC	Mínimo: Máximo: Média

Tabela 2: Métricas MOOD

Métrica	Valor encontrado
MIF	
AIF	
MHF	
AHF	
PF	
CF	

Orientações:

1. Métricas MOOD:

Avaliação única para todo o sistema.

Valor máximo que pode ser atingido: 1,0.

Tabela 3: Parâmetros avaliados pelo conjunto de Métricas MOOD

Parâmetros	Métricas MOOD					
	MIF	AIF	MHF	AHF	PF	CF
Reutilização	X	X	X	X	X	X
Capacidade de teste	X	X			X	X
Manutenção	X	X	X	X	X	X
Eficiência	X	X				
Funcionalidade	X	X	X	X		
Segurança			X	X		
Propensão a erros						X

2. Métricas CK:

Avaliação de cada classe do sistema:

Não há valor máximo a ser atingido.

Tabela 4: Parâmetros avaliados pelo conjunto de Métricas CK

Parâmetros	Métricas CK					
	DIT	NOC	CBO	RFC	LCOM	WMC
Capacidade de teste	X	X	X	X		
Eficiência	X	X				
Reutilização	X	X	X			X
Manutenção	X	X	X	X		X
Falha no projeto das classes					X	
Tempo de desenvolvimento						X
Propensão a erros	X	X	X	X	X	X

3. Como proceder a avaliação:

(a) O conjunto MOOD faz a avaliação geral do sistema e o conjunto CK ajuda a identificar a(s) classe(s) com problemas.

(b) Fazer aplicação periódica das métricas.

(c) Guardar o histórico dos resultados das métricas.

(d) A cada aplicação, verificar:

  i. Se problemas anteriores foram corrigidos.

  ii. Se surgiram novos problemas.

(e) Para a avaliação do projeto, incluir os dados da página 3.